microelectronics group

Lucent Technologies
Bell Labs Innovations

Evaluation Kit for USS-720 *Instant USB*™ USB-to-*IEEE** 1284 Bridge

Introduction

The Lucent Technologies Microelectronics Group USS-720 Evaluation Kit is an assembly of all the peripheral-related hardware, software, and documentation necessary to evaluate the USS-720 device and begin development efforts. The kit is comprised of three primary components:

- Information Manual that contains a collection of data sheets, application notes, and other documents regarding the USS-720, its device driver, and sample applications
- In-System Design (ISD) USB Smart Cable
- A 3 1/2 in. diskette containing related device drivers and sample application software for use with OSR2.1/QFE 1214 and *Microsoft Windows*† 98.

A complete description of the contents is listed in the Evaluation Kit Contents section.

* *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.
† *Microsoft* and *Windows* are registered trademarks of Microsoft Corporation.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Table of Contents (continued)

Evaluation Kit Contents

The Lucent Technologies USS-720 Evaluation Kit consists of the following components:

1. One USS-720 Information Manual containing:
   - An introduction to the USS-720 Evaluation Kit
   - USS-720 Software Licensing information covering the related software
   - Lucent Technologies *Incorporating Customer Data into USS-720 Evaluation Kit Software Using the Build Me One Utility*, Application Note, February 1999, Rev. 1 (AP99-001CMPR-01)
   - Lucent Technologies *USS-720* Instant USB *USB-to-*IEEE *1284 Bridge*, Preliminary Data Sheet, September 1999, Rev. 5 (DS98-393CMPR-5)
   - Lucent Technologies *Typical Circuit Showing USS-720 Bridging USB to Parallel Port*, Application Note, February 1999, Rev. 2 (AP97-069CMPR-2)
   - Lucent Technologies USS-720: *USB Device Driver*, Preliminary User Guide, February 1999, Rev. 2 (MN98-005CMPR-02)
   - Lucent Technologies USS-720: *USB Port Monitor*, Application Note, February 1999, Rev. 2 (AP98-004CMPR-02)
   - Lucent Technologies *Instructions for Downloading Software for Use with the In-System Design USB Smart Cable*, Application Note, June 1999, Rev. 3 (AP97-070CMPR-03)
2. One ISD USB Smart Cable: This cable assembly contains a Lucent Technologies USS-720 device embedded in the Centronics "B" connector end.
3. One 3 1/2 in. diskette containing the following files:

■ glucent.zip	// Zipped text file containing example description of bytes used to program an external EEPROM to be used with Lucent USS-720. **Note:** Prior to programming the EEPROM, the data in this sample file must be modified by the customer to include the customer's specific data.
■ CleanUp.exe	// The Cleanup application is a file unistall program to be used prior to re-installing driver files during the USS-720 USB enumeration process. Cleanup.exe is a utility provided as a convenience to developers using the evaluation kit. It removes the driver files from the system. Cleanup.exe only works with the driver files provided with the evaluation kit and is not suitable, nor licensed, for distribution to end-users.
■ \Win95_OSR21:	
— USS720.INF	// ISD USB Smart Cable Install file
— USS720CI.DLL	// ISD USB Smart Cable PnP Class Installer
— USS720.SYS	// Release version of USS-720 USB Device Driver for OSR2.1
— USS720IN.DLL	// ISD USB Smart Cable PnP Printer Port Monitor Installer
— USS720MN.DLL	// Release version of ISD USB Printer Port Monitor
■ \Win95_OSR21\Debug:	
— USS720.INF	// Debug version of USS720.INF for OSR2.1
— USS720.SYS	// Debug version of USS720.SYS for OSR2.1
— USS720CI.DLL	// Debug version of USS720CI.DLL for OSR2.1
— USS720IN.DLL	// Debug version of USS720IN.DLL for OSR2.1
— USS720IO.H	// Include file required when developing an interface to the USS-720 device driver
— USS720MN.DLL	// Debug version of ISD USB Printer Port Monitor

Evaluation Kit Contents (continued)

- \Win98:

— USS720.INF	// ISD USB Smart Cable Install file
— USS720.SYS	// Release version of USS-720 USB Device Driver for *Windows* 98
— USS720CI.DLL	// ISD USB Smart Cable PnP Class Installer
— USS720IN.DLL	// ISD USB Smart Cable PnP Printer Port Monitor Installer
— USS720MN.DLL	// Release version of ISD USB Printer Port Monitor

- \Win98\Debug

— USS720.INF	// Debug version of USS720.INF for *Windows* 98
— USS720.SYS	// Debug version of USS720.SYS for *Windows* 98
— USS720CI.DLL	// Debug version of ISD USB Smart Cable PnP Class Installer
— USS720IN.DLL	// Debug version of USS720IN.DLL for *Windows* 98
— USS720IO.H	// Include file required when developing an interface to the USS-720 device driver
— USS720MN.DLL	// Debug version of ISD USB Printer Port Monitor

- \SAMPLES

— MakeFile	// MakeFile file required for building
— test.c	// Sample source
— test.exe	// Executable version of sample source
— test.rc	// Resource file required for building
— Sources	// Source file required for building

Hardware Requirements

In addition to the hardware included in the USS-720 Evaluation Kit, the following hardware is required in order to use the kit:

- Personal computer with 486 (or higher) processor with:
  - — 16 Mbytes of RAM.
  - — USB connector.

  And one of the two following operating systems:

  - — *Windows* 98 or
  - — *Windows* 95 version 4.00.950b with the USB supplement (QFE 1214), also known as OSR2.1.

Note: No upgrade path is available from previous versions of *Windows* 95 version 4.00.950 to OSR2.1; *Windows* 95 version 4.00.950B (OSR2.0) must first be installed, and then the system can be upgraded to OSR2.1.

- Printer or other peripheral device conforming to *IEEE* Standard 1284.

Getting Started Instructions

An initial evaluation of the USS-720 device can be performed using the In-System Design USB Smart Cable (i.e., the Lucent USS-720-based USB printer cable), the above-mentioned required hardware, and software supplied on the 3 1/2 in. diskette included in this kit. The basic evaluation procedure consists of the following steps outlined below for OSR2.1 and *Windows* 98:

OSR2.1 Cable Installation Instructions:

1. Connect the USB printer cable to a printer that is turned on and follow the instructions listed below. (Note that although it is not necessary to plug the USB printer cable into a printer to perform the following steps, the use of the printer provides a better demonstration of the USS-720 device's functionality.)
2. With the PC running OSR2.1/QFE 1214, plug the USB end of the USB printer cable into the PC's USB port.

   A New Hardware Found box will appear stating that an Unknown Device has been found. Then, the Building Driver Information Database box will appear as the database is being built. At the end of this process, the Update Device Driver Wizard box will appear on the screen.
3. Follow the prompts to complete the installation of the Unknown Device, i.e., the USB printer cable:
   a. Insert the diskette into the floppy disk drive.
   b. Click on the Next button in the Update Device Driver Wizard box. Using the “Browse” button, direct the system to the Win95_OSR21 folder on the diskette to find the driver for the “Lucent USS720-based USB Cable.”
   c. Click on the Finish button.
4. At this point, the files should automatically be copied from the diskette. The installation process is now complete.

Note: If a Plug and Play printer is being used, then a Plug and Play prompt may appear at this point to install the printer drivers.

5. To verify that the Lucent USS-720-based USB cable has been correctly installed, right click on My Computer, and then left click on Properties to access the System Properties box.
6. Choose the Device Manager tab in this box. The “Lucent USS720-based USB Cables” device type should appear in the list under “Computer”.
7. Double click on “Lucent USS720-based USB Cables”. Another entry, “Lucent USS720-based USB Cable” device will appear underneath.
8. Now double click on “Lucent USS720-based USB Smart Cable”. A “Lucent USS720-based USB Cable Properties” box will appear. If the installation was successful, the device status will state that the device is working properly. If the installation was not successful, use the Cleanup.exe file on the diskette to first uninstall the software, then return to step 2 above.

Note: Cleanup.exe is a utility provided as a convenience to developers using the evaluation kit. It removes the driver files from the system. Cleanup.exe only works with the driver files provided with the evaluation kit and is not suitable, nor licensed, for distribution to end-users.

If the installation was successful and a printer was attached to the USB printer Cable in step 1, a test page can now be printed:

- Click on Start/Settings/Printers.
- Right click on the icon for the printer connected to the USB printer cable, then left click on Properties.
- Choose the Details tab. Select the correct USBLPT port [e.g., USBLPT1 (USB Port)] under “Print to the following port:”. Under “Print using the following driver:”, select the driver that corresponds to the correct printer. Click on the Apply button and then select the “General” tab.
- To begin the printing process, click on the Print Test Page button.

Getting Started Instructions (continued)

***Windows* 98 Cable Installation Instructions:**

1. Connect the USB printer cable to a printer that is turned on and follow the instructions listed below. (Note that although it is not necessary to plug the USB printer cable into a printer to perform the following steps, the use of the printer provides a better demonstration of the USS-720 device's functionality.)
2. With the PC running *Windows* 98, plug the USB end of the USB printer cable into the PC's USB port.

   A New Hardware Found box will appear stating that an Unknown Device has been found.
3. Use the Wizard to complete the installation of the Unknown Device, i.e., the Lucent USS-720-based USB printer cable:
   a. You will be prompted to insert the diskette into the floppy disk drive. Insert the Evaluation Kit’s diskette into the floppy drive and click OK.
   b. You will be prompted to direct the operating system to the correct location of the drivers. Using the “Browse” button, direct the system to the Win98 folder on the diskette, and click OK.
   c. Follow the prompts and the software files should automatically be copied from the diskette. The installation process is now complete.

Note: If a Plug and Play printer is being used, then a Plug and Play prompt may appear at this point to install the printer drivers.

4. To verify that the USB printer cable has been correctly installed, right click on “My Computer”, and then left click on Properties to access the System Properties box.
5. Choose the Device Manager tab in this box. The “Lucent USS720-based Cables” device type should appear in the list under “Computer”.
6. Double click on “Lucent USS720-based USB Cables”. Another entry, “Lucent USS720-based USB Cable” will appear underneath.
7. Now double click on “Lucent USS720-based USB Cable”. This will cause the “Lucent USS720-based USB Cable Properties” box to appear. If the installation was successful, the device status will state that the device is working properly. If the installation was not successful, use the Cleanup.exe file on the diskette to first uninstall the software, then return to step 2 above.

Note: Cleanup.exe is a utility provided as a convenience to developers using the evaluation kit. It removes the driver files from the system. Cleanup.exe only works with the driver files provided with the evaluation kit and is not suitable, nor licensed, for distribution to end-users.

If the installation was successful and an already-installed printer was attached to the USB printer cable in step 1, a test page can now be printed:

- Click on Start/Settings/Printers.
- Right click on the icon for the printer connected to the USB Cable, then left click on Properties.
- Choose the Details tab. Select the correct USBLPT port under “Print to the following port:”. Under “Print using the following driver:”, select the driver that corresponds to the correct printer. Click on the Apply button and then select the General tab of the box.
- To begin the printing process, click on the Print Test Page button.

If further assistance is required during the installation and/or printing process, contact either Lucent Technologies Applications Engineering or In-System Design (see the *USB Applications Support* document in this manual).

Software Updates

In addition to the files included on the evaluation kit’s diskette, newer versions of the software may be available on In-System Design’s Web Site: **http://www.in-system.com**. The library, uss720_dev, contains files that can be used to evaluate the USS-720 device.

To access software files from In-System Design's home page, follow these steps:

1. Click on Drivers.
2. In the box labeled Library Name, type the library name: **uss720_dev**
3. Type the password for the library: **usb_to_lpt**
   (**Note:** Use all lower-case letters.)

Notes for Developers

Please note that the driver files in this evaluation kit are for evaluation purposes only. Developers can obtain free drivers suitable for distribution from the web site at www.in-system.com. Selecting "Drivers", filling out the information in the "Build Me One" section and then selecting the "Submit" button will generate a request for drivers to In-System Design. Select "Frequently Asked Questions" to obtain more details about the required fields such as Vendor ID. If the information is entered correctly, a response will be sent for verification within 2—5 working days.

Use of the USS-720 Evaluation Kit by developers implies a basic understanding of the USB and *IEEE* 1284 interfaces as well as basic competency in board design and *Windows* driver programming. The information provided in this kit is intended only as a guide for the development of a USB-to-*IEEE* 1284 bridging solution and, as such, is not warranted for suitability to any particular purpose.

Following are additional notes regarding the software provided with the USS-720 Evaluation Kit:

USB Driver and Port Monitor

The USS-720 is supplied with a USB driver and a USB port monitor. The port monitor allows *Windows* printer drivers to print to an *IEEE* 1284-compliant printer connected to the USS-720. Note that the supplied software will work only with printer drivers that use the *Windows* spooling system to interface with the USS-720 USB port monitor. See the *USS-720 USB Port Monitor* Application Note in this Manual for details.

Plug and Play

The Plug and Play capabilities of the supplied software are limited to the enumeration of the USS-720 device only.

Port Creation

The USS-720 software driver supports printing via the creation of a port named USBLPTx, where x is an integer between 1 and 127. Each instantiation of the driver will generate another USBLPTx port. If two developers both have products based on the USS-720 connecting to the same USB bus, it may not be clear which device is connected to which USBLPTx port.

Notes for Developers (continued)

EEPROM

Developers using the USS-720 must use an external serial EEPROM (or the equivalent) in their design and create their own hex data file for use in programming the EEPROM at their site. See the USS-720 data sheet and the glucent.zip file on the kit's diskette for more information. The configuration data stored in this serial EEPROM is used by the *Microsoft* host software during enumeration to load the appropriate drivers. Using unique identification data in the EEPROM provides a means for the developer to ensure that only their software is loaded for use with their device. Otherwise, the enumeration of a camera using the USS-720 could cause the software for a USS-720-based printer to be loaded, resulting in a system which does not function correctly.

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

USB Applications Support

Applications support for Lucent Technologies USS-720 Universal Serial Bus products can be obtained by contacting the following:

Lucent Technologies Microelectronics Group

1247 S. Cedar Crest Blvd.
Allentown, PA 18103 U.S.A.
Phone: (610) 712-2947
FAX: (610) 712-2820, Attention: USB Applications Engineering
e-mail: usb@lucent.com

77 Science Park Drive #03-18
Cintech III, Singapore 118256
Phone: (65) 870 5733
FAX: (65) 777 7495, Attention: Jeffrey Lam
e-mail: zpjlam@lucent.com

In-System Design, Inc.

12426 W. Explorer Drive
Suite 100
Boise, Idaho 83713 U.S.A.
Phone: (208) 377-9222
FAX: (208) 377-9333
e-mail: support@in-system.com   (Subject: USB)

Please see “Notes for Developers” in the Introduction of this Information Manual for more information on available driver support.

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

USS-720
Software Use Agreement

Evaluation Kit Software

Introduction

In accordance with the license agreement that follows, your use of the software and other information furnished as part of the USS-720 Evaluation Kit is intended for uses in connection with evaluating and testing Lucent's USS-720 device and beginning development efforts. In addition, subject to the license agreement, you may also distribute the software with the your USS-720 based peripheral. The software furnished as part of the Kit, along with updates, is also available (without charge) at In System Design, Inc.'s website at www.in-system.com.

Before evaluating or distributing the software, it is advisable to visit the website to access any updates and in some cases, to furnish certain information relating to your peripheral.

BY USING THE SOFTWARE FURNISHED AS PART OF THE EVALUATION KIT OR ACCESSING THE SOFTWARE AT THE IN-SYSTEM WEBSITE, YOU AGREE TO THE LICENSE AGREEMENT THAT FOLLOWS. IF YOU DO NOT AGREE WITH THE LICENSE AGREEMENT PLEASE RETURN THE KIT TO LUCENT AND REFRAIN FROM ANY USE OF THE SOFTWARE.

Please Note: The software available through the website that is free of charge is suitable for most printing applications where the end-user manually installs the printer driver. However, the free software has limited functionality with respect to some features, such as "Child Discovery" which enables the attached printer to "Plug and Play." Such enhanced versions of the software are available under a separate licensing agreement based on per unit royalties payable to In-System Design, Inc. (Contact In-System Design Marketing for more details: jack@in-system.com).

USS-720 Driver Software (Object Code) License Agreement

Note: Capitalized terms have the meanings ascribed to them in the Definitions Appendix.

The use and distribution of USS720 driver software by any LICENSEE is strictly limited to the following terms and conditions:

ARTICLE I—RIGHT TO USE LICENSED SOFTWARE

1.01 Grant of Right

(a) LUCENT TECHNOLOGIES INC. ("LUCENT") grants to LICENSEE personal, nontransferable, and nonexclusive rights (i) to use the LICENSED SOFTWARE to evaluate and test the LUCENT DEVICE, (ii) to use LICENSED SOFTWARE with the LUCENT DEVICE when incorporated in SYSTEMS and (iii) subject to Section 1.01(e), to furnish copies of LICENSED SOFTWARE to LICENSEE's customers worldwide in object-code form solely for use in SYSTEMS.

(b) No right is granted for the use of LICENSED SOFTWARE for or by any third person except as provided above or for use of any portion of LICENSED SOFTWARE other than for use in SYSTEMS.

USS-720 Driver Software (Object Code) License Agreement (continued)

(c) LICENSEE may make those copies of LICENSED SOFTWARE necessary to the use by LICENSEE for which rights are granted hereunder, provided that each such copy contains any copyright and/or proprietary notice appearing on or in the LICENSED SOFTWARE being copied.

(d) LICENSEE agrees that it will not use or copy LICENSED SOFTWARE except as authorized herein.

(e) LICENSEE agrees that any distribution of the LICENSED SOFTWARE to customers shall only be in object code form and pursuant to a license agreement containing essentially the following:

- (i) only a personal, nontransferable, and nonexclusive right to use such copy in SYSTEMS is granted to such customer;
- (ii) no ownership interest in LICENSED SOFTWARE is transferred to such customer;
- (iii) such customer will not copy LICENSED SOFTWARE except as necessary to use LICENSED SOFTWARE in SYSTEMS and for backup and archive purposes in connection with such use and will include all copyright and/or proprietary notices in such copies;
- (iv) if a customer's right-to-use is terminated for any reason, such customer will either destroy or return all copies of LICENSED SOFTWARE in its possession;
- (v) such customer will not export or re-export LICENSED SOFTWARE without the appropriate Unites States and/or foreign government licenses;
- (vi) such customer will not reverse compile or disassemble LICENSED SOFTWARE, subject to applicable exceptions if such customer’s address is in one of the Member States of the European Economic Community; and
- (vii) LUCENT does not warrant LICENSED SOFTWARE, does not assume any liability regarding LICENSED SOFTWARE, and does not undertake to furnish any support or information to such customer regarding LICENSED SOFTWARE.

(f) LICENSEE agrees to use its best efforts to enforce the agreements with customers referred to in Section 1.01(e).

(g) The agreement specified in Section 1.01(e) may be (i) a written agreement signed by the customer or (ii) a written agreement on or accompanying the package containing the LICENSED SOFTWARE that is visible or otherwise disclosed to the customer before the customer opens the package, that the customer accepts by opening the package and that complies with applicable law governing such agreements.

1.02 Furnishing of LICENSED SOFTWARE

Subject to receipt by LUCENT of the fee specified in Section 2.01, within a reasonable time after such receipt, LUCENT shall furnish LICENSED SOFTWARE to LICENSEE in the form specified in the attached Schedule.

1.03 Ownership

No ownership interest in LICENSED SOFTWARE is transferred to LICENSEE hereunder. LICENSEE's ownership interest is limited solely to those portions of LICENSEE's adaptations that do not contain any of LICENSED SOFTWARE.

1.04 U.S. Export Control

LICENSEE acknowledges that LICENSED SOFTWARE is subject to export restrictions under U.S. Export Administration Regulations and international arrangements of the U.S. Government. LICENSEE agrees not to ship or otherwise transmit LICENSED SOFTWARE except in compliance with all such U.S. Government regulations and arrangements.

USS-720 Driver Software (Object Code) License Agreement (continued)

ARTICLE II—FEES AND PAYMENTS

2.01 Fees

LICENSEE shall, within thirty (30) days after execution of this Agreement by both parties and invoice by LUCENT, pay to LUCENT a fee of _________ for the rights granted herein.

2.02 Taxes

LICENSEE shall pay any tax (and any related interest or penalty), however designated, imposed as a result of the existence or operation of this agreement, including any tax which LICENSEE is required to withhold or deduct from payments to LUCENT, except (i) any such tax imposed upon LUCENT or any of its subsidiaries in the jurisdiction in which the aforesaid office of LICENSEE is located if such tax is allowable as a credit against United States income taxes of LUCENT or any of its subsidiaries; and (ii) any income tax imposed upon LUCENT or any of its subsidiaries by the United States or any governmental entity within the United States proper (the fifty states and the District of Columbia). To assist LUCENT in obtaining the credit identified in (i) of this Section 2.03, LICENSEE shall furnish LUCENT with such evidence as may be required by United States taxing authorities to establish that any such tax has been paid. Fees specified in this agreement do not include taxes.

ARTICLE III—TERMINATION

3.01 Termination for Breach

If LICENSEE fails to fulfill one or more of its obligations under this agreement, LUCENT may, upon its election and in addition to any other remedies that it may have, at any time terminate all the rights granted by it hereunder by not less than two (2) months' written notice to LICENSEE specifying any such breach, unless within the period of such notice all breaches specified therein have been remedied. Upon such termination, LICENSEE shall destroy all copies of LICENSED SOFTWARE in its possession and certify such destruction in writing to LUCENT within thirty (30) days. In the event of such termination by LUCENT, LUCENT shall have no obligation to refund any amount paid pursuant to Section 2.01.

3.02 Survival

The obligations of LICENSEE under Sections 1.04 and 4.04 shall survive and continue after any termination of rights under this agreement.

USS-720 Driver Software (Object Code) License Agreement (continued)

ARTICLE IV—MISCELLANEOUS PROVISIONS

4.01 Agreement Prevails

This agreement shall prevail notwithstanding any conflicting terms or legends which may appear on or in LICENSED SOFTWARE.

4.02 Warranty and Indemnity

(a) LUCENT warrants that LICENSED SOFTWARE will be in good working order at the time it is furnished. If LICENSED SOFTWARE is not in good working order at such time, LUCENT will, upon return of LICENSED SOFTWARE at any time within ninety (90) days after LICENSED SOFTWARE was first furnished, replace LICENSED SOFTWARE without charge.

(b) The devices in the LUCENT CHIP SET are covered by the warranty and indemnity provisions set forth in the LUCENT terms and conditions pursuant to which such products are sold.

(c) LUCENT AND ITS AFFILIATES MAKE NO OTHER REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, REGARDING LICENSED SOFTWARE. BY WAY OF EXAMPLE, BUT NOT OF LIMITATION, LUCENT AND ITS AFFILIATES MAKE NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR THAT THE USE OF LICENSED SOFTWARE WILL NOT INFRINGE ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. LUCENT AND ITS AFFILIATES SHALL NOT BE HELD TO ANY LIABILITY WITH RESPECT TO ANY CLAIM BY LICENSEE, OR A THIRD PARTY ON ACCOUNT OF, OR ARISING FROM, THE USE OF LICENSED SOFTWARE.

(d) Nothing herein confers on LICENSEE, or upon anyone claiming under LICENSEE, any license (expressly, impliedly, by estoppel or otherwise) under any patent of LUCENT or others covering or relating to any combination in which LICENSED SOFTWARE is or might be used.

4.03 Nothing Construed

Nothing contained herein shall be construed as:

(i) conferring by implication, estoppel, or otherwise any license or right to use any name, trade name, trademark, service mark, symbol, or any other identification or any abbreviation, contraction, or simulation thereof;

(ii) an obligation upon LUCENT or any of its affiliates to furnish any person, including LICENSEE, any assistance of any kind whatsoever or any information or documentation other than LICENSED SOFTWARE to be furnished pursuant to Section 1.02; or

(iii except for the right to furnish object-code versions to customers as provided in Section 1.01(a)(ii), a right to sell, lease, sublicense or otherwise transfer or dispose of LICENSED SOFTWARE, in whole or in part.

4.04 Confidentiality

(a) LICENSEE agrees to hold all parts of LICENSED SOFTWARE in confidence for LUCENT. LICENSEE further agrees not to make any disclosure of LICENSED SOFTWARE (including methods or concepts utilized therein) to anyone, except to employees of LICENSEE to whom such disclosure is necessary to the use for which rights are granted hereunder.

(b) LICENSEE shall ensure that an obligation not to disclose confidential information forms part of its terms of employment.

USS-720 Driver Software (Object Code) License Agreement (continued)

(c) LICENSEE's obligations under this Section 4.04 shall not apply to any information relating to LICENSED SOFTWARE (including any method or concept utilized therein) that:

(i) is or becomes available without restriction to the general public by acts not attributable to LICENSEE or its employees,

(ii) was rightfully in LICENSEE's possession without limitation on disclosure before disclosure hereunder to LICENSEE,

(iii) is rightfully disclosed to LICENSEE by a third party without restrictions on disclosure, or

(iv) is independently developed by LICENSEE.

4.05 Publicity

LICENSEE agrees that it will not, without the prior written consent of LUCENT;

(i) use in advertising, publicity, packaging, labeling, or otherwise any trade name, trademark, trade device, service mark, symbol, or any identification or any abbreviation, contraction or simulation thereof owned by LUCENT or any of its affiliates or used by LUCENT or any of its affiliates to identify any of its or their products or services or

(ii) represent, directly or indirectly, that any product or service of LICENSEE is a product or service of LUCENT or any of its affiliates or is made in accordance with or utilizes any information or documentation of LUCENT or any of its affiliates.

4.06 Nonassignability

Neither this Agreement nor any rights hereunder shall be assignable or transferable (in insolvency proceedings or otherwise) by LICENSEE without the express written consent of LUCENT. Any purported transfer or assignment in contravention of this provision shall be considered void and ineffective.

4.07 Addresses

Any statement, notice, request, or other communication hereunder shall be deemed to be sufficiently given to the addressee and any delivery hereunder deemed made when sent by certified mail addressed as follows:

to LUCENT:

LUCENT TECHNOLOGIES INC.
555 Union Boulevard
Allentown, Pa. 18103-1229
ATTN.:

to LICENSEE:

{company name}
{company street address}
{company city, state, zip}
{company phone number}

or to such changed address as the addressee shall have specified by written notice.

4.08 Integration

This agreement sets forth the entire agreement and understanding between the parties as to the subject matter hereof and merges all prior discussions between them. Neither of the parties shall be bound by any warranties, understandings, or representations with respect to such subject matter other than as expressly provided herein, in the object code agreement or in a writing executed with or subsequent to the execution of this agreement by an authorized representative of the party to be bound thereby.

USS-720 Driver Software (Object Code) License Agreement (continued)

4.09 Applicable Law

The construction and performance of this agreement shall be governed by the laws of the State of New York, U.S.A., excluding choice of law rules.

4.10 Disputes

If a dispute arises out of or relates to this Agreement, or its breach, the parties agree to submit the dispute to a sole mediator selected by the parties or, at any time at the option of a party, to mediation by the American Arbitration Association (“AAA”). If not thus resolved it shall be referred to a sole arbitrator selected by the parties within thirty (30) days of the mediation or, in the absence of such selection, to AAA binding arbitration which shall be governed by the United States Arbitration Act and judgment on the award may be entered in any court having jurisdiction. The mediation or arbitration shall be held in New York City. The arbitrator may not limit, expand or otherwise modify the terms of the Agreement. The arbitrator shall not have authority to award punitive, exemplary or other damages in excess of compensatory damages and each party irrevocably waives any claim thereto. Each party shall bear its own expenses and those of the mediator and arbitrator shall be borne equally. The parties, their representatives, other participants and the mediator and arbitrator shall hold the existence, content and result of mediation and arbitration in confidence.

IN WITNESS WHEREOF, each of the parties has caused this agreement to be executed in duplicate originals by its duly authorized representatives on the respective dates entered below.

LUCENT TECHNOLOGIES INC.

By	{Marketing person's name}
Title	{Marketing person's title}
Date	

Signed:____________________________________

{company name}

By	
Title	
Date	

Signed:____________________________________

USS-720 Driver Software (Object Code) License Agreement (continued)

DEFINITIONS APPENDIX

LUCENT DEVICE means the LUCENT USS720 device.

LICENSEE means a customer of Lucent's USS720 device using the LICENSED SOFTWARE.

LICENSED SOFTWARE means all or any portion of the software files in object-code form, other information and documentation specifically listed in the attached "Schedule for LUCENT USS720 Driver Software and any updates that may be furnished to LICENSEE."

SYSTEM means any system manufactured by LICENSEE incorporating one or more LUCENT DEVICES.

Schedule for LUCENT USS720 Driver Software (Object Code)

1. Software Files

The following software files will be furnished in object-code form on 3.5" Floppy Disk or by electronic mail:

----

1. Class Installer,	USS720CI.DLL;
2. USB WDM device driver,	USS720.SYS;
3. Port monitor installer,	USS720IN.DLL;
4. Port monitor,	USS720MN.DLL;
5. Install file,	USS720.INF;

2. Documentation

One copy of the following document(s) will be furnished:

1. USS720 USB Device Driver
2. USS720 USB Port Monitor

Application Note, Rev. 1
February 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

Incorporating Customer Data into USS-720 Evaluation Kit Software Using the Build Me One Utility

Introduction

Using the Build Me One utility found on In-System Design's web site, the USS-720 Evaluation Kit software can be modified to include data specific to a customer's USB peripheral. This document describes the components, customization data, and other general topics related to the free software provided by Lucent Technologies and In-System Design in support of the USS-720 *Instant USB*™ device. See the instructions under Customer-Defined Data below for information on obtaining the free software.

Software Components

The files obtained by using the Build Me One utility are the software necessary for standard PnP installation and operation of the USS-720-based USB Smart Cable. This software consists of:

1. Class Installer,	xxxxxxCI.DLL;
2. WDM device driver,	xxxxxxSB.SYS;
3. Port monitor installer,	xxxxxxIN.DLL;
4. Port monitor,	xxxxxxMN.DLL;
5. Install file,	xxxxxxSB.INF;

The six-character "xxxxxx" strings are defined by the customer. (Refer to 'Driver Name' information in the Customer-Defined Data section below.)

Example: If "XYZPRT" is the 'Driver Name' defined by the customer, then the Class Installer software file would be named XYZPRTCI.DLL.

Class Installer

The Class Installer checks that the operating system version is compatible with the USB Smart Cable software being loaded. This is necessary as there are two different binary versions of this software, one for *Microsoft Windows** 95/OSR2.1 and one for *Microsoft Windows* 98. Furthermore, the Class Installer checks that all required components are installed for OSR2.1 systems (*Windows* 95 with the USB supplement and QFE).

If the Class Installer finds the operating system is correct, the device driver and port monitor are installed. If the Class Installer finds the operating system either does not correspond to the software or the operating system does not have all required components, an error message is displayed.

WDM Device Driver

The device driver for the USB Smart Cable follows the *Windows* Driver Model specification. Note there are different binaries for *Windows* 95 and *Windows* 98 as different libraries are linked. Note that the supplied software will work only with printer drivers that use the *Windows* spooling system to interface with the USS-720 USB port monitor.

Port Monitor

The port monitor, developed by In-System Design, is responsible for the communication between the *Windows* spooler and a printer. This port monitor enables printing from *Windows* applications using the USB Smart Cable. The port monitor controls the I/O port to which the physical printer is connected and is responsible for the communication channel between the spooler and the printer.

Install File

The install file is modified to match the information provided by the customer. The information in the external EEPROM is used during the Plug and Play process to load the correct device driver.

* *Microsoft* and *Windows* are registered trademarks of Microsoft Corporation.

Customer-Defined Data

Customer-defined data may be submitted to In-System Design using the Build Me One utility located on the In-System Design web site. Access **www.in-system.com**; select the Drivers area and then the utility labeled “Build Me One.”

The modification of the software requires the following strings to be provided by the customer:

1. Device Descriptor String
2. Manufacturer String
3. Vendor ID
4. Product ID
5. Driver Name
6. Port Name

Device Description String

This string describes the name of the customer’s product. The install file will include a product name string of 20 characters (including spaces).

Example: “XYZ PRINTER CABLE”.

Manufacturer String

This string is the name of the manufacturer and is limited to 20 characters (including spaces).

Example: “XYZ CORP.”

Vendor ID

A unique USB Vendor ID is required for the software modification process. The Vendor ID field is limited to four hex characters. Prior to submitting data to In-System Design, the customer must obtain a unique Vendor ID from the USB organization [see **www.usb.org/developers**]. Refer to the Frequently Asked Questions section at the bottom of the In-System Design “Drivers” web page for more information on Vendor IDs.

Example: “0x12AB”

Product ID

The Product ID typically represents the revision number of the product. The Product ID field is limited to four characters. The revision number appears in a dialog box of the device properties. The most and least significant bytes are swapped so that a Product ID of 0x0100 is displayed as revision 1.00. The customer is responsible for selecting its own Product ID.

Example: “0x0100”

Driver Name

The driver name is the six-character prefix that is used to form the customer’s software file names described in the Software Components section found earlier in this document.

Example: “XYZPRT”

Port Name

The port name is the six-character name that identifies the LPT port which represents the USB Smart Cable attached device. In the Evaluation Kit version of the software, this port is USBLPTx: (where x is a number between 1 and 127).

Example: “LPTXYZ”

Plug and Play Operation (PnP)

Installation of the USB Smart Cable software is started by connecting the USB Smart Cable to the host PC or USB hub. Enumeration occurs and the proper software components for the USB Smart Cable are loaded through the normal PnP operation. Note that there is no user interface or installer program that the user needs to execute. The user will be prompted to insert the floppy containing the USB Smart Cable driver during the PnP operation.

Printer Enumeration

The printer attached to the USB Smart Cable does not enumerate with this software.

Operating Systems Supported

This software runs in *Windows* 95, and *Windows* 98.

Windows 95 must actually be OSR2.1 version 1214, which contains the USB supplement and QFE. Each vendor who purchases the USB Smart Cable from In-System Design must execute an agreement with *Microsoft* in order to distribute the USB supplement. *Microsoft* may license the vendor to distribute the following files to update *Windows* 95 software:

1. OEMUSB.EXE version 4.03.1214,
2. DETROITR.EXE,
3. USBUPD1.EXE version 4.03.1214.

To execute a distribution agreement with *Microsoft*, contact Stephanie Selden at *Microsoft*, email **sselden@microsoft.com**.

Note: As of August 1998, *Microsoft* is no longer licensing the OSR2.1 supplement.

Power Management

The USB Smart Cable supports power management on the PC if its operating system supports the specific power management feature. The USB Smart Cable supports suspend and resume functions in all supported operating systems.

Note that OSR2.1 supports suspend and resume functions but there is no USB support in OSR2.1 for “0 volt suspend” in which case some PCs cut the power upon suspending. *Windows* 98 does support “0 volt suspend.”

USB Hardware Types Supported

The USB Smart Cable operates with all OHCI and UHCI USB controllers and hubs that are compliant with the USB Specification, Revision 1.0.

Installation Process

The software for the USB Smart Cable is loaded automatically by *Windows* PnP operation.

USBLPTx Ports

Evaluation kit version of software:

For each USB Smart Cable plugged into a host’s USB, a port will be added called USBLPTx (where x can be from 1 through 127). These ports will remain in the system with their current status if a printer is associated with the port. If no printer is associated with a USBLPT port when the corresponding USB Smart Cable is removed, the port is removed from the system.

Modified version of the USS-720 software:

For each USB Smart Cable plugged into a host’s USB, a port will be added called XXXXXXx (where x can be from 1 through 127). The “XXXXXX” string is defined by the customer, for example: “LPTXYZ”.

EEPROM

Developers using the USS-720 must use an external serial EEPROM (or the equivalent) in their design and create their own hex data file for use in programming the EEPROM at their site. See the USS-720 Data Sheet and the glucent.zip file on the kit’s diskette for more information. The configuration data stored in this serial EEPROM is used by the *Microsoft* host software during enumeration to load the appropriate drivers. Using unique identification data in the EEPROM provides a means for the developer to ensure that only their software is loaded for use with their device.

Printer Types Supported

The following printers have been thoroughly tested and are known to work with the USS-720 software. Some printers require bidirectional mode to be disabled. There are known to be some printers which do not work with the USB Smart Cable. This is often due to an incompatibility with the printer's *Windows* driver. Some printers also have incompatibilities in their implementation of the *IEEE* 1284 printer port. Lucent's USS-720 customer assumes all responsibility for testing the printers that will be used by their customers.

Note also that this list of printers represents only those printers that In-System Design has used to extensively test the USS-720 software. There are many additional printers that work with the USS-720 and its associated software.

Table 1. Print Types Supported

Printer	Vendor	Comments
Bubble Jet BJ[1]-200ex InkJet printer	*Canon*[1]	Bidirectional Disabled
Bubble Jet BJC[1]-240 InkJet printer	*Canon*	Bidirectional Disabled
Bubble Jet BJC-610 InkJet printer	*Canon*	Bidirectional Disabled
Bubble Jet BJC-4200 Color InkJet printer	*Canon*	Bidirectional Disabled
Epson Stylus[2] 500 Color InkJet printer	*Epson*[2] America	—
Epson Stylus 800 Color InkJet printer	*Epson* America	—
DeskJet[3] 340CM printer	*Hewlett-Packard*[3]	—
DeskJet 672C InkJet printer	*Hewlett-Packard*	—
DeskJet 692C InkJet printer	*Hewlett-Packard*	—
DeskJet 694C InkJet printer	*Hewlett-Packard*	—
DeskJet 722C InkJet printer	*Hewlett-Packard*	—
DeskJet 820Cse InkJet printer	*Hewlett-Packard*	Bidirectional Disabled
DeskJet 870Cxi InkJet printer	*Hewlett-Packard*	—
DeskJet 1200C InkJet printer	*Hewlett-Packard*	—
LaserJet[3] 4si	*Hewlett-Packard*	—
LaserJet 5P	*Hewlett-Packard*	—
LaserJet 6P	*Hewlett-Packard*	—
LaserJet 6L PCL	*Hewlett-Packard*	—
1000 ColorFine	*Lexmark*[4]	—
SuperScript 860 laser printer	*NEC*[5]	—

1. *BJ*, *BJC*, and *Canon* are registered trademarks and *Bubble Jet* is a trademark of Canon Inc.
2. *EPSON*, the EPSON logo, and EPSON Stylus, are registered trademarks of Seiko Epson Corporation, registered in the U.S. and other countries.
3. *Hewlett-Packard*, *DeskJet*, and *LaserJet* are registered trademarks of Hewlett-Packard Company.
4. *Lexmark* is a registered trademark of Lexmark International, Inc.
5. *NEC* is a registered trademark of NEC Technologies Inc.

Preliminary Data Sheet, Rev. 5
September 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

USS-720 *Instant USB*™ USB-to-*IEEE** 1284 Bridge

Features

Device Features:

- Full compliance with the *Universal Serial Bus Specification Revision 1.0*
- On-chip transceivers for USB
- Low power consumption allows part to be powered from USB connection
- Dual on-chip USB packet buffers for fast response
- Fully compatible USB host device drivers available
- Fully compliant with USB printer device class specification
- Implemented in Lucent Technologies Microelectronics Group's 0.35 µm, 3 V standard-cell library
- Complete single-chip solution in a 44-pin MQFP
- Evaluation kit available

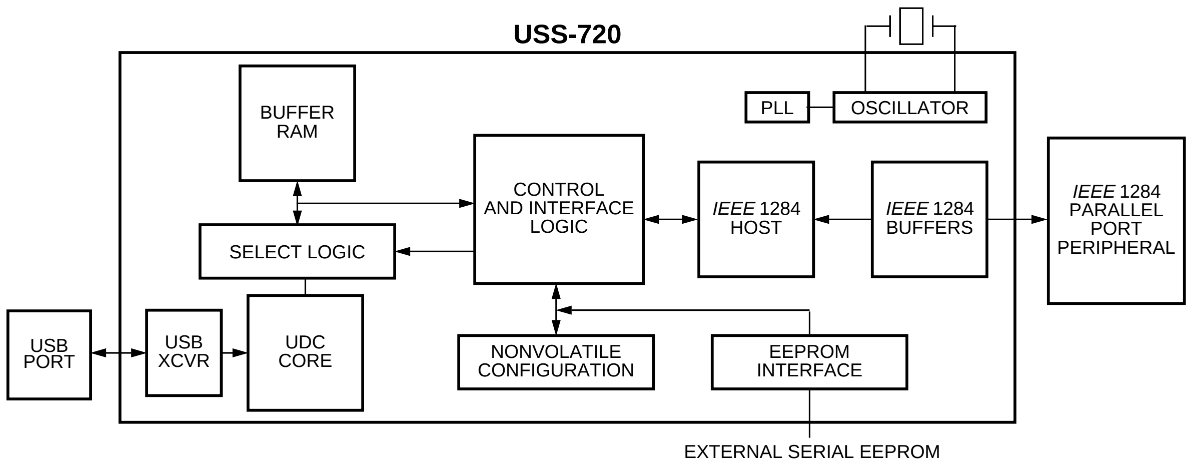
IEEE 1284 (Parallel) Features:

- Transparent, fully automatic support for true bidirectional communication
- Hardware initiates and manages automatic negotiation for the fastest protocol available
- Supports standard PC parallel port register-based operation
- Support of multiple logical channels
- Maximum throughput: 1.216 Mbytes/s (ECP mode)

Description

The USS-720 integrated circuit connects an *IEEE* 1284 parallel port peripheral to the universal serial bus (USB). It is designed to be a low-cost, single-chip embedded solution requiring minimal external components. It is suitable for a wide range of applications, from integrated applications where the IC is mounted on a printed-circuit board inside a product, to stand-alone applications where the chip provides a standard parallel port to a USB-capable computer. The USS-720 software included in the Evaluation Kit allows peripheral vendors to evaluate and test the USS-720 device, which is embedded in the *Centronics*† end of a USB-to-parallel port cable provided with the kit. This solution requires no firmware changes on the parallel port peripheral.

* *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.
† *Centronics* is aregistered trademark of Centronics Data Computer Corporation.



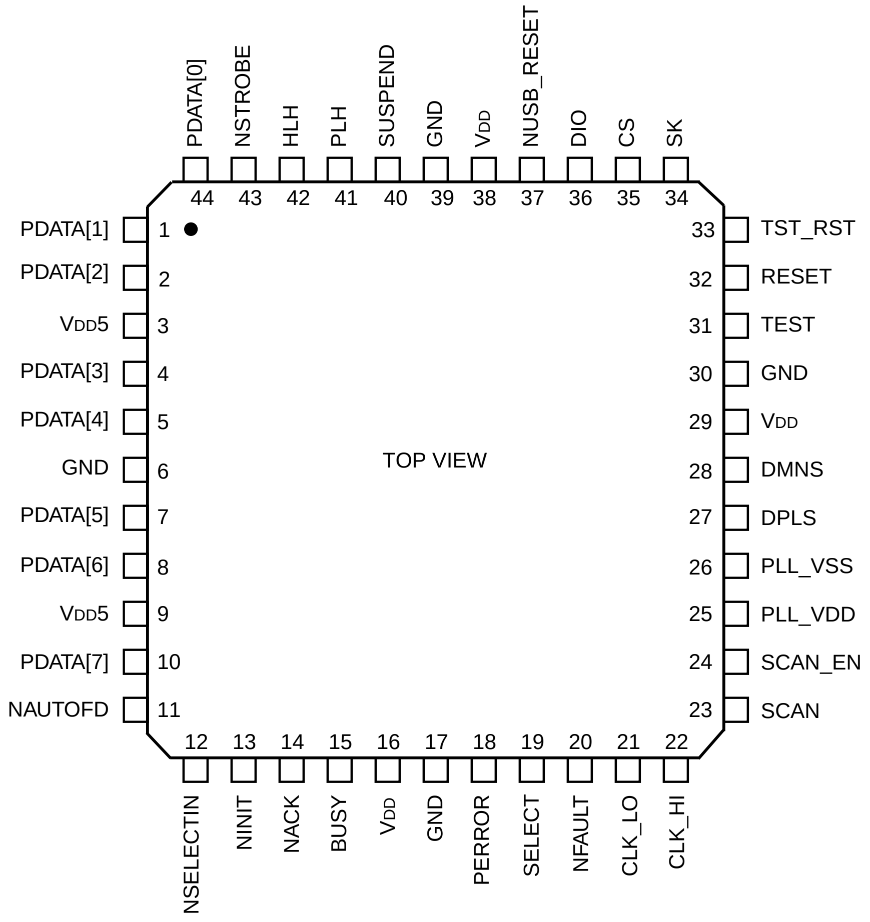
5-5109a.r6

Figure 1. Block Diagram

Note: Advisories are issued as needed to update product information. When using this data sheet for design purposes, please contact your Lucent Technologies Microelectronics Group Account Manager to obtain the latest advisory on this product.

Table of Contents

Pin Information



Note: An N before symbol names indicates active-low.

Figure 2. Pin Diagram

Table 1. Pin Descriptions

Pin	Symbol*	Type	Name/Description
1	PDATA[1]	I/O	**Parallel Port Data Signal Bit 1.**
2	PDATA[2]	I/O	**Parallel Port Data Signal Bit 2.**
3	VDD5	P	**5 V Power Supply for 5 V Parallel Port Signals.**
4	PDATA[3]	I/O	**Parallel Port Data Signal Bit 3.**
5	PDATA[4]	I/O	**Parallel Port Data Signal Bit 4.**
6	GND	P	**Ground.**
7	PDATA[5]	I/O	**Parallel Port Data Signal Bit 5.**
8	PDATA[6]	I/O	**Parallel Port Data Signal Bit 6.**

* An N before symbol names indicates active-low.

Pin Information (continued)

Table 1. Pin Descriptions (continued)

Pin	Symbol*	Type	Name/Description
9	VDD5	P	**5 V Power Supply for 5 V Parallel Port Signals.**
10	PDATA[7]	I/O	**Parallel Port Data Signal Bit 7.**
11	NAUTOFD	O	**Parallel Port nAutoFd Signal (Active-Low).**
12	NSELECTIN	O	**Parallel Port nSelectIn Signal (Active-Low).**
13	NINIT	O	**Parallel Port nInit Signal (Active-Low).**
14	NACK	I	**Parallel Port nAck Signal (Active-Low).**
15	BUSY	I	**Parallel Port Busy Signal.**
16	VDD	P	**3.3 V Power Supply.**
17	GND	P	**Ground.**
18	PERROR	I	**Parallel Port PError Signal.**
19	SELECT	I	**Parallel Port Select Signal.**
20	NFAULT	I	**Parallel Port nFault Signal (Active-Low).**
21	CLK_LO	I	**Clock Low.** Crystal or CMOS input.
22	CLK_HI	O	**Clock High.** Crystal or no connection.
23	SCAN	I	**Scan.** This signal is only used for production testing. Tie to GND for normal operation.
24	SCAN_EN	I	**Scan Enable.** When high, internal chip scan is enabled. This signal is only used for production testing. Tie to GND for normal operation.
25	PLL_VDD	P	**3.3 V Analog Power Supply for PLL.**
26	PLL_VSS	P	**Analog Ground for PLL.**
27	DPLS	I/O	**USB DPLS Signal.**
28	DMNS	I/O	**USB DMNS Signal.**
29	VDD	P	**3.3 V Power Supply.**
30	GND	P	**Ground.**
31	TEST	I	**Test.** This signal is only used for production testing. Tie to GND for normal operation.
32	RESET	I	**Reset.** This signal is only used for production testing. Tie to GND for normal operation.
33	TST_RST	I	**Test Reset.** This signal is only used for production testing. Tie to GND for normal operation.
34	SK	O	**Serial ROM Clock.**
35	CS	O	**Serial ROM Chip Select.**
36	DIO	I/O	**Serial ROM Data Signal.**
37	NUSB_RESET	O	**USB Reset (Active-Low).** Indicates USB reset condition.
38	VDD	P	**3.3 V Power Supply.**
39	GND	P	**Ground.**
40	SUSPEND	O	**Suspend.** Indicates USB bus in suspend.
41	PLH	I	**Parallel Port Peripheral Logic High Signal.**
42	HLH	O	**Parallel Port Host Logic High Signal.**
43	NSTROBE	O	**Parallel Port nStrobe Signal (Active-Low).**
44	PDATA[0]	I/O	**Parallel Port Data Signal.**

* An N before symbol names indicates active-low.

Overview

The USS-720 creates a bridge between one USB port and one *IEEE* 1284 enhanced parallel port. Internally, the USS-720 contains an integrated USB transceiver, a USB device controller (UDC) core, an *IEEE* 1284 core, integrated *IEEE* 1284 buffers, storage for USB configuration data, data buffers, and control logic to tie the blocks together. The USS-720 also contains an onboard oscillator, PLL, and reset block for single-chip operation.

In use, the USB port of the USS-720 is connected via a USB cable to a host computer or the downstream port of a USB hub. Host software sends commands and data to the USS-720 and receives status and data from the USS-720 using the USB protocol.

The *IEEE* 1284 enhanced parallel port of the device is connected to a peripheral device. If the peripheral is *IEEE* 1284 compatible, then the associated features and communication modes can be used. The USS-720 provides both automatic and manual operation of the *IEEE* 1284 port.

USB Port

The USB port on the USS-720 is electrically and logically compliant with the *USB Specification Revision 1.0*.

Device Descriptor, Configurations, and Interfaces

Supported Descriptors

- **Device**.
- **Configuration**.
- **Interface**. The USS-720 device supports one interface with three alternate settings.
  - Interface 0, alternate settings 0 and 1 are compliant with the *USB Device Class Definition for Printing Devices, Release Candidate 1.0*.
  - Interface 0, alternate setting 2 is a vendor-specific interface.
- **Endpoint**. The USS-720 supports the following endpoints:
  - Control endpoint. Accessible as endpoint 0 in all three alternate interface settings.
  - Bulk Out endpoint. Accessible as endpoint 1 in all three alternate interface settings.
  - Bulk In endpoint. Accessible as endpoint 2 in alternate interface settings 1 and 2.
  - Interrupt endpoint. Accessible as endpoint 3 in alternate interface setting 2.
- **String**.

Descriptor Locations

Descriptor data is supplied from an external ROM or other device. The USS-720 provides support for 93CS56 and 93CS66 EEPROM interfaces. (**Note:** Substitution EEPROM components must be pin and functional compatible with the 93CS56L/66L. 93C56L/66L, 93CS46L, and 93C46L EEPROM parts will **not** function correctly with the USS-720.) The format for the externally supplied descriptor data requires that the descriptors loaded be preceded by the total length of the descriptor to be returned. In the case of the device descriptor, this value would be 0x12, which is redundant since the descriptor returned is always 0x12 bytes long; the first byte of the descriptor would also be 0x12. The length of the configuration descriptor, however, is not the same as the first byte of that descriptor, since the configuration descriptor and all associated interface and endpoint descriptors are returned as a whole.

The USS-720 also contains a set of device, configuration, interface, and endpoint descriptors that may be used in development and prototyping. Retrieval of the onboard descriptors will occur if no external descriptor data is supplied.

USB Port (continued)

The format for the externally supplied data is as shown in Table 2. The addressing for the specified EEPROM device is word aligned, so the following restrictions are placed upon the starting locations for the configuration and string descriptors.

- The configuration descriptor must start at word address 0x13 (byte address 0x26).
- String descriptors must start at an address that is aligned on a double-word boundary.

Table 2. Externally Supplied Data Format

Byte Address	Word Address	Value
0x00	0x00	0x12
0x01—0x12	—	Device descriptor as defined in *USB Specification Revision 1.0*.
0x26	0x13	Total length of string to be returned in response to a GET_CONFIGURATION_DESCRIPTOR request.
0x27—0x2F	—	Configuration descriptor as defined in *USB Specification Revision 1.0*.
0x30—???*	—	Interface and endpoint descriptors to be returned in response to a GET_CONFIGURATION_DESCRIPTOR request.
Any even word address beyond the end of the configuration descriptor return string.		Length of string to return in response to a GET_STRING_DESCRIPTOR request. This address divided by two should be included in the low byte of the wIndex field of the GET_DESCRIPTOR standard command.
Next address—end of string descriptor.		String descriptor as defined in *USB Specification Revision 1.0*.

* The question marks (???) indicate that this byte address value is determined for the user's application based on the above information. The number is going to vary depending on how the user formats their configuration information.

Note: Tables 3—14 describe the descriptor data contained in the USS-720's internal ROM. This information can be used as a guide in creating the externally supplied descriptor data.

Onboard Device Descriptor

There is only one device descriptor for each USB device. This descriptor contains the definitions of the device class and the device subclass, among other things.

Table 3. Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	0x12	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x01	Device descriptor type.
2	bcdUSB	Word	0x0100	USB Specification release number in BCD.
4	bDeviceClass	Byte	0x00	Interface specific. See Tables 6, 8, and 11.
5	bDeviceSubClass	Byte	0x00	Interface specific. See Tables 6, 8, and 11.
6	bDeviceProtocol	Byte	0x00	Interface specific. See Tables 6, 8, and 11.
7	wMaxPacketSize0	Byte	0x08	Maximum packet size for endpoint 0.
8	idVendor	Word	0x047E	Vendor ID for Lucent Technologies.
10	idProduct	Word	0x1001	Product ID.
12	bcdDevice	Word	0x0103	Device release number in BCD.
14	iManufacturer	Byte	0x00	Index of string descriptor describing manufacturer.
15	iProduct	Byte	0x00	Index of string descriptor describing this product.
16	iSerialNumber	Byte	0x00	Index of string descriptor describing the device's serial number.
17	bNumConfigurations	Byte	0x01	Number of possible configurations.

USB Port (continued)

Configuration Descriptor

The USS-720 has one default configuration descriptor. This descriptor has one interface, which has three alternate settings. The three alternate settings and the endpoints that they support are shown in Table 4.

Table 4. Alternate Settings and Supported Endpoints

Endpoint	Interface		
	Alternate Setting 0	**Alternate Setting 1**	**Alternate Setting 2**
Control Pipe	Endpoint Number 0 8 bytes	Endpoint Number 0 8 bytes	Endpoint Number 0 8 bytes
Bulk Out Pipe	Endpoint Number 1 64 bytes	Endpoint Number 1 64 bytes	Endpoint Number 1 64 bytes
Bulk In Pipe	—	Endpoint Number 2 64 bytes	Endpoint Number 2 64 bytes
Interrupt Pipe	—	—	Endpoint Number 3 4 bytes

Table 5. Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	0x09	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x02	Configuration descriptor type.
2	bTotalLength	Word	0x004E	Number of bytes in this configuration. This includes the configuration descriptor plus all of the interface and endpoint descriptors.
4	bNumInterfaces	Byte	0x01	The USS-720 has one interface.
5	bConfigurationValue	Byte	0x01	Value to use as an argument to Set Configuration to select this configuration.
6	iConfiguration	Byte	0x00	Index of string descriptor describing this configuration.
7	bmAttributes	Byte	0x80	Configuration characteristics: **Bit** **Description** **USS-720** 7 Bus-powered. Set. 6 Self-powered. Cleared. 5 Remote wakeup. Cleared. 4—0 Reserved, set to 0. Cleared.
8	MaxPower	Byte	0x31	Maximum power consumption of this configuration. Units are mA ∗ 2; therefore, the value 0x31 is equivalent to 98 mA.

USB Port (continued)

Interface Descriptors

The USS-720 supports a single interface with three alternate settings.

Interface 0, Alternate Setting 0 (I0:A0)

Table 6. Interface Descriptor, I0:A0

Offset	Field	Size	Value	Description
0	bLength	Byte	0x09	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x04	Interface descriptor type.
2	bInterfaceNumber	Byte	0x00	Zero-based value identifying the number of this interface.
3	bAlternateSetting	Byte	0x00	Value used to select this alternate interface.
4	bNumEndpoints	Byte	0x01	Number of endpoints used by this descriptor.
5	bInterfaceClass	Byte	0x07	Printer class.
6	iInterfaceSubClass	Byte	0x01	Printer subclass.
7	bInterfaceProtocol	Byte	0x01	Unidirectional interface.
8	iInterface	Byte	0x00	Index to string describing this interface.

Table 7. Bulk Out Endpoint Descriptor, I0:A0:E1

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x01	This is an Out endpoint, endpoint number 1.
3	bmAttributes	Byte	0x02	This is a Bulk endpoint.
4	wMaxPacketSize	Word	0x0040	Maximum data transfer size.
6	bInterval	Byte	0x00	Does not apply to bulk endpoints.

USB Port (continued)

Interface 0, Alternate Setting 1 (I0:A1)

Table 8. Interface Descriptor, I0:A1

Offset	Field	Size	Value	Description
0	bLength	Byte	0x09	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x04	Interface descriptor type.
2	bInterfaceNumber	Byte	0x00	Zero-based value identifying the number of this interface.
3	bAlternateSetting	Byte	0x01	Value used to select this alternate interface.
4	bNumEndpoints	Byte	0x02	Number of endpoints used by this descriptor.
5	bInterfaceClass	Byte	0x07	Printer class.
6	iInterfaceSubClass	Byte	0x01	Printer subclass.
7	bInterfaceProtocol	Byte	0x02	Bidirectional interface.
8	iInterface	Byte	0x00	Index to string describing this interface.

Table 9. Bulk Out Endpoint Descriptor, I0:A1:E1

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x01	This is an Out endpoint, endpoint number 1.
3	bmAttributes	Byte	0x02	This is a Bulk endpoint.
4	wMaxPacketSize	Word	0x0040	Maximum data transfer size.
6	bInterval	Byte	0x00	Does not apply to Bulk endpoints.

Table 10. Bulk In Endpoint Descriptor, I0:A1:E2

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x82	This is an In endpoint, endpoint number 2.
3	bmAttributes	Byte	0x02	This is a Bulk endpoint.
4	wMaxPacketSize	Word	0x0040	Maximum data transfer size.
6	bInterval	Byte	0x00	Does not apply to Bulk endpoints.

USB Port (continued)

Interface 0, Alternate Setting 2 (I0:A2)

Table 11. Interface Descriptor, I0:A2

Offset	Field	Size	Value	Description
0	bLength	Byte	0x09	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x04	Interface descriptor type.
2	bInterfaceNumber	Byte	0x00	Zero-based value identifying the number of this interface.
3	bAlternateSetting	Byte	0x02	Value used to select this alternate interface.
4	bNumEndpoints	Byte	0x03	Number of endpoints used by this descriptor.
5	bInterfaceClass	Byte	0xFF	Vendor specific.
6	iInterfaceSubClass	Byte	0x00	—
7	bInterfaceProtocol	Byte	0xFF	Vendor specific.
8	iInterface	Byte	0x00	Index to string describing this interface.

Table 12. Bulk Out Endpoint Descriptor, I0:A2:E1

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x01	This is an Out endpoint, endpoint number 1.
3	bmAttributes	Byte	0x02	This is a Bulk endpoint.
4	wMaxPacketSize	Word	0x0040	Maximum data transfer size.
6	bInterval	Byte	0x00	Does not apply to Bulk endpoints.

Table 13. Bulk In Endpoint Descriptor, I0:A2:E2

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x82	This is an In endpoint, endpoint number 2.
3	bmAttributes	Byte	0x02	This is a Bulk endpoint.
4	wMaxPacketSize	Word	0x0040	Maximum data transfer size.
6	bInterval	Byte	0x00	Does not apply to Bulk endpoints.

Table 14. Interrupt Endpoint Descriptor, I0:A2:E3

Offset	Field	Size	Value	Description
0	bLength	Byte	0x07	Size of this descriptor in bytes.
1	bDescriptorType	Byte	0x05	Endpoint descriptor type.
2	bEndpointAddress	Byte	0x83	This is an In endpoint, endpoint number 3.
3	bmAttributes	Byte	0x03	This is an Interrupt endpoint.
4	wMaxPacketSize	Word	0x0004	Maximum data transfer size.
6	bInterval	Byte	0x01	This pipe should be serviced every frame.

USB Port (continued)

Pipes

Four pipes are defined: Control, Bulk Out, Bulk In, and Interrupt.

Control Pipe

The Control pipe is the default pipe, used for USB setup and control packets. Its maximum packet size is 8 bytes. The Control pipe is also used for class- and vendor-specific commands that:

- Configure class- and vendor-specific features.
- Retrieve Device, Configuration, and String descriptors.

Note: Descriptor data shares the physical buffer used to transfer Bulk In data. Retrieving this data will result in the loss of any reverse channel data currently in the Bulk In buffer.

- Read and write the parallel port registers.
  — Access standard parallel port register set.
  — Read/write an address byte from/to the peripheral in EPP Mode.
  — Read/write a data byte from/to the peripheral in EPP Mode (but multiple bytes can be transferred more efficiently via the Bulk Out pipe).
- Read Printer Class Get Device ID data.

Note: This data shares the physical buffer used to transfer Bulk In data. Retrieving this data will result in the loss of any reverse channel data currently in the Bulk In buffer.

Bulk In Pipe

The Bulk In pipe is used to read data bytes from the peripheral in Automatic Mode and register-based ECP Mode. Its maximum packet size is 64 bytes. The buffer used for this pipe is shared with the Control pipe for descriptor data and *IEEE* 1284 Device ID string data. The Control pipe has priority over Bulk In data, so any data in this buffer when a request is made for descriptor data or *IEEE* 1284 Device ID string will be lost.

Bulk Out Pipe

The Bulk Out pipe is used to send data to the peripheral in Automatic Mode and in Compatibility, EPP, or ECP Register Modes. Its maximum packet size is 64 bytes.

Interrupt Pipe

The Interrupt pipe is used to report changes in parallel port and buffer status to the host. Interrupt packets are 4 bytes in length. When the Interrupt pipe is enabled by host software, the host automatically polls the USS-720 once per frame. The USS-720 returns 4 bytes of status whenever an interrupt condition exists, as described in the Interrupts section on page 23, and returns nothing otherwise. This enables the host to detect and react to parallel port and buffer status changes without explicit polling.

Interpipe Synchronization

With commands and data going to different pipes, and data potentially being buffered inside the USS-720, it could be difficult for host software to maintain serialization of operations on the peripheral. This can be done by reading the registers to determine the status of the USS-720. Or, the Interrupt pipe status mechanism described above can be used to alleviate this problem. Software can use the port status and buffer status information thus returned to determine when buffered data has been sent and when port control commands have been processed and it is safe to continue. Since interrupt information is returned to the software automatically and only when it changes, overhead for the host operating system and driver software is kept low when using the interrupt pipe as opposed to polling the registers.

Requests

The USS-720 can respond to three different types of requests:

- Standard USB device requests.
- Class-specific requests.
- Vendor-specific requests.

Standard Requests

The USS-720 supports all of the standard device requests described in Chapter 9, Device Framework, of the USB Specification except Set Descriptor:

- Clear Feature.
- Get Configuration.
- Get Descriptor. Direct requests for interface and endpoint descriptors are not supported in the USS-720 and will cause the Control pipe to be stalled.
- Get Interface.
- Get Status.
- Set Address.
- Set Configuration.
- Set Interface.
- Set Feature.

USB Port (continued)

Printer Class-Specific Requests

Printer class-specific requests supported by the USS-720 are listed in Table 15.

Table 15. Class-Specific Requests

Label	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GET_DEVICE_ID	10100001B	0	Config Index	Interface and Alternate Setting	Maximum Length	*IEEE* 1284 Device ID String
GET_PORT_STATUS	10100001B	1	Zero	Interface	1	BYTE
SOFT_RESET	00100011B	2	Zero	Interface	Zero	[None]

GET_DEVICE_ID

This request returns an *IEEE* 1284 Device ID string. This command is supported by all three alternate interface settings.

The Device ID is a length field followed by a case-sensitive string of ASCII characters. The first 2 bytes are the length of the sequence, including the two length bytes. The first byte is the most significant; length values of 0000h, 0001h, and 0002h are reserved.

Following the two length bytes, the sequence is composed of a series of keys and values of the form:

key: value {, value};

repeated for each key. As indicated, each key will have one value, and may have more than one value. The minimum necessary keys are MANUFACTURER, COMMAND SET, and MODEL (case sensitive). These keys may be abbreviated as MFG, CMD, and MDL. Each key (and each value) is a string of characters. Any characters except colon (:), comma (,), and semicolon (;) may be included as part of the key or value string. Any leading or trailing white space in the string is ignored by the parsing program, but is still counted as part of the overall length of the sequence.

For more details, see *IEEE* 1284-1994, Section 7.6.

Note: The USS-720 satisfies this request by requesting Device ID data from the attached *IEEE* 1284 peripheral. If the peripheral does not support Device ID, the USS-720 will return a zero-length data packet to the host.

GET_PORT_STATUS

This request returns the current status of the printer. Table 16 defines the data returned and describes the format of the status data. This command is only supported by the two printer class-specific alternate interface settings (0 and 1). Attempts to issue this command to interface 0, alternate setting 2 will result in a stall.

Table 16. Get Port Status Data

Bit	Description
7—6	Reserved, will always read 0.
5	Paper error.
4	Select.
3	Not error: 0 = error, 1 = no error.
2—0	Reserved, will always read 0.

SOFT_RESET

This request flushes all buffers and resets the Bulk Out and Bulk In pipes to their default states, and also resets all parallel port hardware and registers to their default state. This command is supported in all three alternate interface settings.

USB Port (continued)

Vendor-Specific Requests

Vendor-specific requests supported by the USS-720.

Table 17. Vendor-Specific Requests

Label	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GET_1284_REGISTER	11000000B	3	Address	Zero	7	Register Data
SET_1284_REGISTER	01000000B	4	Address Data	Zero	Zero	[None]

GET_1284_REGISTER

The high byte of the wValue field specifies the address of the register that is to be read. The USS-720 responds by returning the current values in all of the parallel port registers. This command is only supported in interface 0, alternate setting 2. Attempts to issue this command to alternate settings 0 or 1 will result in a stall.

SET_1284_REGISTER

The wValue field specifies the address of the parallel port register to be written in the high byte and the value to be written in the low byte. This command is only supported in interface 0, alternate setting 2. Attempts to issue this command to alternate settings 0 or 1 will result in a stall.

IEEE 1284 Port

The *IEEE* 1284 port on the USS-720 is compliant with the *IEEE* 1284-1994 standard. The parallel port operates in two distinct modes. In fully Automatic Mode, the *IEEE* 1284 protocol is implemented completely in hardware. Compatibility Mode, Nibble Mode, and ECP Mode (with or without RLE compression) are supported, with all negotiation, termination, and other features of the protocol handled transparently by the hardware. The USS-720 also features a Register Mode, which presents a standard register interface to the host. These two modes provide the host with two distinct operating paradigms. In Automatic Mode, the host software interacts with the USS-720 as if with a USB-capable printer; while in Register Mode, the USS-720 emulates standard PC parallel port hardware. In the Automatic Mode, the host application software doesn't know that the USB data stream is being converted to *IEEE* 1284 protocol. In the Register Mode, it need not know that its interaction with parallel port registers is actually talking place remotely over a USB link.

Automatic mode is recommended for almost all applications. (**Note**: The drivers included in the Lucent Technologies USS-720 Evaluation Kit use only Automatic Mode.) Automatic Mode implements all negotiation handshakes automatically for Compatibility, Nibble, and ECP modes. In Register mode, the user must do the negotiations manually in software. Register mode can be useful when implementing a nonstandard parallel port interface.

Register-Based Operation

In its Register Mode of operation, the USS-720 emulates standard host-side parallel port hardware, with the register accesses being performed remotely over a USB connection. As in the standard register set, the interface mode is controlled by the Mode field in the Extended Control Register. The supported modes and their meanings are given in the Extended Control Register section on page 5-20, and operation and required software interaction for each of the supported modes are described in the sections that follow.

Standard Mode (000)

In this mode, the parallel port is under full software control, with no form of hardware assist. Software has complete control of all parallel port signals. This mode can be used for negotiations, terminations, proprietary handshake sequences, etc. As in standard host-side parallel port hardware, the parallel port data lines are unidirectional outputs in this mode.

Bidirectional Mode (001)

This mode is identical to Standard Mode (000), except that the direction of the parallel port data lines may be controlled with the Direction bit in the Control Register.

Compatibility Mode (010)

This mode provides hardware-based Compatibility Mode data transfers. Data sent to the USS-720 over the Bulk Out pipe will be transferred automatically to the peripheral using Compatibility Mode.

ECP Mode (011)

This mode provides hardware-based ECP Mode data transfers. To use ECP, the host software should negotiate for ECP Mode via the Control and Status Registers, then set Mode to 011. At this time, the Compress Enable bit in the USS-720 Control Register should also be set appropriately.

EPP Mode (100)

This mode provides hardware-based EPP Mode data transfers. To use EPP, the host software should negotiate for EPP Mode via the Control and Status Registers, and then set Mode to 100.

EPP mode in the USS-720 has the following data transfer rate characteristics. Note that the rates are approximate.

Table 18. Transfer Rates

Direction	UHCI	OHCI
Forward	1 Mbyte/s	1 Mbyte/s
Reverse	250 bytes/s	1 Kbyte/s

Because the reverse channel operation is implemented in the USB Control Pipe, the transfer rates are limited according to the particular implementation of the host controller, either UHCI or OHCI.

The high performance of the forward direction in EPP mode makes it attractive for implementations requiring little reverse channel traffic.

IEEE 1284 Port (continued)

Registers

Nine parallel port registers are available to the host. They are read and written using the GET_1284_REGISTER and SET_1284_REGISTER vendor-specific commands described above. The SET_1284_REGISTER writes a value into a particular register. Writes may either affect the configuration of the hardware or have a direct effect on parallel port control lines. In the case of the EPP Registers, writes initiate EPP write cycles on the parallel port.

The GET_1284_REGISTER returns seven register values: Status, Control, Extended Control, USS-720 Control, Data, EPP Address/Data, and USS-720 Setup Registers, in that order (see 19). Register values can also be read via the Interrupt pipe, which returns the values of Status, Control, Extended Control, and USS-720 Control Registers, in that order (see Table 20). Reads may affect the contents of the registers in one of two ways. Any register read clears any interrupt status that may exist at the time of the read. Also, a read targeting the EPP address or EPP Data Register will initiate the appropriate EPP read cycle on the parallel port. The value returned will be the address or data byte read from the peripheral as a result of that read cycle.

Table 19. GET_1284_REGISTER Data

Byte	Register Data
0	Status
1	Control
2	Extended Control
3	USS-720 Control
4	Data
5	EPP Address/Data
6	USS-720 Setup

Table 20. Interrupt Pipe Read Data

Byte	Register Data
0	Status
1	Control
2	Extended Control
3	USS-720 Control

Each of the nine registers are described in detail in Tables 21—29.

IEEE 1284 Port (continued)

Data Register

Table 21. Data Register

Data Register							**Address: 0**	
Bit	7	6	5	4	3	2	1	0
Symbol	D7	D6	D5	D4	D3	D2	D1	D0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Symbol	Bit Description
7—0	D7—D0	**Data.** This register is equivalent to and operates in the same manner as the Data Register in a standard host-side parallel port controller chip. The register is writable when Auto Mode is 0 and the Mode field in the Extended Control Register is set to 000 or 001. It is always readable. The read value will be the value of the data latched into the register unless the Mode field is set to 001 and the Direction bit in the Control Register is set to 1 (Input Mode). In this case, the read value will be the value present on the parallel port data lines.

Status Register

Table 22. Status Register

Status Register							**Address: 1**	
Bit	7	6	5	4	3	2	1	0
Symbol	nBusy	nAck	PError	Select	nFault	—	PLH	Timeout
Access	Read	Read	Read	Read	Read	Read	Read	Read
Default	X	X	X	X	X	X	X	0

Bit	Symbol	Bit Description
7	nBusy	**Inverted Busy.** An inverted version of the parallel port Busy signal.
6	nAck	**Parallel Port nAck Signal.**
5	PError	**Parallel Port PError Signal.**
4	Select	**Parallel Port Select Signal.**
3	nFault	**Parallel Port nFault Signal.**
2	—	**Reserved.**
1	PLH	**Peripheral Logic High.** The parallel port PLH signal.
0	Timeout	**EPP Time-Out.** This bit indicates that a time-out has occurred during an EPP read or write. If the peripheral fails to respond to an EPP read or write for longer than 10 µs, this bit will be set and an interrupt will be returned if interrupts are enabled. This bit is cleared by a read.

IEEE 1284 Port (continued)

Control Register

Table 23. Control Register

Control Register						Address: 2		
Bit	7	6	5	4	3	2	1	0
Symbol	HLH	EPP mask	Direction	Int enbl	SelectIn	nInit	AutoFd	Strobe
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	1	0	0	1	1	0	0

Bit	Symbol	Bit Description
7	HLH	**Host Logic High.** The parallel port HLH signal.
6	EPP mask	**EPP Time-Out Interrupt Mask.** This bit masks the generation of an interrupt upon time-out of an EPP data or address transfer. Note that in typical host-side parallel port controller chips, this interrupt condition is grouped with and controlled by the Interrupt Enable bit in this register; so for exact emulation of typical parallel port hardware, this bit should always be written with the inverse of the Interrupt Enable bit.
5	Direction	**Parallel Port Direction.** When the Mode field in the Extended Control Register is set to 001, this bit controls the direction of the parallel port data lines. When set to 0, the lines are in Output Mode, and when set to 1, they are in Input Mode (see the Data Register on previous page). This bit also controls the direction of the interface in ECP Mode (011). It has no effect in Modes 000 or 010 (which are unidirectional only), or 100 (where the direction is uniquely determined by the type of access to the EPP Registers).
4	Int enbl	**Interrupt Enable.** This bit enables interrupt generation on nAck events. If this bit is set, interrupt status will be generated on transitions of nAck from low to high (this status being reflected by the nAck Interrupt bit in the USS-720 Control Register).
3	SelectIn	**Inverted nSelectIn.** An inverted version of the parallel port nSelectIn signal.
2	nInit	**Parallel Port nInit Signal.**
1	AutoFd	**Inverted nAutoFd.** An inverted version of the parallel port nAutoFd signal.
0	Strobe	**Inverted nStrobe.** An inverted version of the parallel port nStrobe signal.

IEEE 1284 Port (continued)

EPP Address Register

Table 24. EPP Address Register

EPP Address Register							**Address: 3**	
Bit	7	6	5	4	3	2	1	0
Symbol	A7	A6	A5	A4	A3	A2	A1	A0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	X	X	X	X	X	X	X	X

Bit	Symbol	Bit Description
7—0	A7—A0	**EPP Address.** This register is equivalent to and operates in the same manner as the EPP Address Register in a standard host-side parallel port controller chip. The register is writable when Auto Mode is 0 and the Mode field in the Extended Control Register is set to 100. A write to this register initiates an EPP address write transfer on the parallel port. The register is always readable. When the Mode is set to 100, a read access will initiate an EPP address read transfer on the parallel port, and the value returned will be the address value read from the peripheral. Reads when not in Mode 100 will return whatever value has been previously latched, but will not have any effect on the parallel port.

EPP Data Register

Table 25. EPP Data Register

EPP Data Register							**Address: 4**	
Bit	7	6	5	4	3	2	1	0
Symbol	D7	D6	D5	D4	D3	D2	D1	D0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	X	X	X	X	X	X	X	X

Bit	Symbol	Bit Description
7—0	D7—D0	**EPP Data.** This register is equivalent to and operates in the same manner as the EPP Data Register in a standard host-side parallel port controller chip. The register is writable when Auto Mode is 0 and the Mode field in the Extended Control Register is set to 100. A write to this register initiates an EPP data write transfer on the parallel port. The register is always readable. When the Mode is set to 100, a read access will initiate an EPP data read transfer on the parallel port, and the value returned will be the data value read from the peripheral. Reads, when not in Mode 100, will return whatever value has been previously latched, but will not have any effect on the parallel port.

IEEE 1284 Port (continued)

ECP Command Register

Table 26. ECP Command Register

ECP Command Register						Address: 5		
Bit	7	6	5	4	3	2	1	0
Symbol	C7	C6	C5	C4	C3	C2	C1	C0
Access	Write	Write	Write	Write	Write	Write	Write	Write
Default	X	X	X	X	X	X	X	X

Bit	Symbol	Bit Description
7—0	C7—C0	**ECP Command.** This register is equivalent to the ECP Address FIFO Register in a standard host-side parallel port controller chip, but has some different restrictions on its usage. The register is writable when Auto Mode is 0 and the Mode field in the Extended Control Register is set to 011 and there is no ECP data in either the Bulk Out buffers or in the process of being transmitted. Writes to this address in a mode other than 011 will be ignored; writes while in mode 011 and when the hardware is busy will generate a NAK. The value written to this register will be transferred to the peripheral as an ECP command.

***IEEE* 1284 Port** (continued)

Extended Control Register

Table 27. Extended Control Register

Extended Control Register							**Address: 6**	
Bit	7	6	5	4	3	2	1	0
Symbol	Mode[2]	Mode[1]	Mode[0]	nAck interrupt	nFault interrupt	Bulk In interrupt	Bulk In empty	Bulk Out empty
Access	R/W	R/W	R/W	Read	Read	Read	Read	Read
Default	0	0	0	0	0	0	1	1

Bit	Symbol	Bit Description
7—5	Mode[2:0]	**Mode.** In Register Mode (when Auto Mode is 0), this bit controls the mode of the parallel port interface. This field is equivalent to the Mode field in a standard host-side parallel port controller chip. The supported modes are as follows: **Mode[2:0] — Mode — Description** 000 — Standard Mode — Full software control, data lines are output only 001 — Bidirectional Mode — Full software control, data lines are bidirectional 010 — Compatibility Mode — Hardware handshaking 011 — ECP Mode — Software negotiations, hardware data transfers 100 — EPP Mode — Software negotiations, hardware data transfers 101 — Reserved 110 — Reserved 111 — Reserved For more information, see the Register-Based Operation section on page 14.
4	nAck Interrupt	**nAck Interrupt.** This bit will be set when the parallel port nAck signal makes a transition from 0 to 1 while the Interrupt Enable bit in the Control Register is set to 1. Interrupt status is cleared by any register read.
3	nFault Interrupt	**nFault Interrupt.** This bit will be set when the parallel port nFault signal makes a transition from 1 to 0 while the nFault Interrupt Mask bit in the USS-720 Control Register is set to 0. An interrupt will also be generated if the mask bit goes low while nFault is low. Interrupt status is cleared by any register read.
2	Bulk In Interrupt	**Bulk In Interrupt.** This bit will be set when Bulk In data is available and the Bulk In Interrupt Mask bit in the Control Register is set to 0. This allows software to use the interrupt pipe to automatically receive notification of available Bulk In data rather than polling with Bulk In requests. Interrupt status is cleared by any register read.
1	Bulk In Empty	**Bulk In Empty.** This bit will be clear when there is Bulk In data available for reading by the host, and set when there is not.
0	Bulk Out Empty	**Bulk Out Empty.** This bit will be clear when there is Bulk Out data waiting in the buffers or in the process of being transmitted over the parallel port, and set otherwise.

IEEE 1284 Port (continued)

USS-720 Control Register

Table 28. USS-720 Control Register

USS-720 Control Register							Address: 7	
Bit	7	6	5	4	3	2	1	0
Symbol	Discon. int mask	Change int mask	Bulk In int mask	Bulk Out int mask	nFault int mask	Reserved	Compress enable	Auto mode
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	0	1	1

Bit	Symbol	Bit Description
7	Discon. int mask	**Disconnect Interrupt Mask.** This bit masks the generation of an interrupt on the detection of what appears to be a disconnection of the peripheral from the parallel port. Disconnect is detected when the Peripheral Logic High signal changes from 1 to 0, or when all parallel port lines driven by the peripheral are high for a long period of time.
6	Change int mask	**Change Interrupt Mask.** This bit masks the generation of an interrupt on the detection of a transition on any of the parallel port lines driven by the peripheral.
5	Bulk In int mask	**Bulk In Interrupt Mask.** This bit masks the generation of an interrupt when Bulk In data is available for reading by the host.
4	Bulk Out int mask	**Bulk Out Interrupt Mask.** This bit masks the generation of an interrupt when the Bulk Out data path goes empty.
3	nFault int mask	**nFault Interrupt Mask.** This bit masks the generation of an interrupt falling edge of nFault when Mode is set to 011 (ECP Mode).
2	Reserved	**Reserved.** This bit must always be written to 0.
1	Compress enable	**Compress Enable.** This bit enables automatic hardware-based RLE compression of outgoing data, for use in ECP with RLE Mode. If software desires to use this feature, this bit should be set before attempting to send data. This bit **must** be cleared for proper operation in ECP (without RLE) Mode.
0	Auto mode	**Auto Mode.** Setting this bit puts the chip in fully Automatic Mode. When set, USS-720 can handle all communications with the peripheral with no assistance from software, and all registers except the USS-720 Control Register are read only. Clearing this bit enables write access to the other registers and disables all automatic operation, provided that none has yet taken place.

IEEE 1284 Port (continued)

USS-720 Setup Register

Table 29. USS-720 Setup Register

USS-720 Setup Register							**Address: 8**	
Bit	7	6	5	4	3	2	1	0
Symbol	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Filter Software Override	Filter Enable
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	1

Bit	Symbol	Bit Description
7	Reserved	**Reserved.**
6	Reserved	**Reserved.**
5	Reserved	**Reserved.**
4	Reserved	**Reserved.**
3	Reserved	**Reserved.**
2	Reserved	**Reserved.**
1	Filter Software Override	**Filter Software Override.** When this bit is set, software can control the digital filtering of incoming parallel port signals with the Filter Enable bit. When clear, filters are controlled by the pull-up or pull-down stage of the SUSPEND pin. (See Filter Bypass Mode.)
0	Filter Enable	**Filter Enable.** Controls digital filtering of incoming parallel port signals when the Filter Software override bit is set. Setting to 1 enables filtering; clearing this bit disables filtering.

IEEE 1284 Port (continued)

Interrupts

The USS-720 can return interrupt status on the interrupt pipe. Interrupt status may be generated as a result of one of seven separately maskable conditions. Any interrupts that are pending will no longer be pending after a read operation. The individual conditions are described in the sections that follow.

nAck Interrupt

The nAck interrupt is enabled by setting the Interrupt Enable bit in the Control Register. An interrupt will be generated whenever nAck transitions from 0 to 1. Interrupt status is indicated by the nAck Interrupt bit in the Extended Control Register.

EPP Time-Out Interrupt

The EPP time-out interrupt is enabled by setting the EPP Time-Out Interrupt Mask bit in the Control Register to 0. Note that this is a change from typical host-side parallel port hardware, where interrupts on EPP time-out conditions are enabled by the Interrupt Enable bit in the Control Register.

An EPP time-out occurs when the peripheral fails to respond to an EPP handshake within the time allowed by the *IEEE* 1284 specification. If this occurs, there is no reliable way to determine whether the peripheral is still functioning or not, or whether the byte in transit was transferred properly, and it will be up to software to attempt to recover by resetting the connection or some other means.

nFault Interrupt

The nFault interrupt is enabled by setting the nFault Interrupt Mask bit in the USS-720 Control Register to 0. Interrupt status is reported via the nFault Interrupt bit in the Extended Control Register. The interrupt is generated when in ECP Mode and either the nFault line transitions from 1 to 0 or the nFault line is low and the interrupt is unmasked. This may indicate that the peripheral has reverse data to transmit.

Bulk In Interrupt

The Bulk In interrupt is enabled by setting the Bulk In Interrupt Mask bit in the USS-720 Control Register to 0. Interrupt status is reported by the Bulk In Interrupt bit in the Extended Control Register, as well as the Bulk In Empty bit in the same register. This interrupt is generated when there is Bulk In data available for reading by the host. By enabling this interrupt, the host may use the automatic polling of the interrupt pipe to receive notification of incoming data, rather than explicitly polling the Bulk In pipe.

Bulk Out Interrupt

The Bulk Out interrupt is enabled by setting the Bulk Out Interrupt Mask bit in the USS-720 Control Register to 0. Bulk Out empty status is reported via the Bulk Out Empty bit in the Extended Control Register. This interrupt is generated when the Bulk Out data pipeline goes completely empty. By enabling this interrupt, the host may use the automatic polling of the interrupt pipe to be notified of the completion of a data transfer, rather than explicitly polling the Bulk Out Empty bit.

Change Interrupt

The Change interrupt is enabled by setting the Change Interrupt Mask bit in the USS-720 Control Register to 0. There is no Interrupt Status bit associated with the Change Interrupt. This interrupt is generated when any of the parallel port signal lines driven by the peripheral (nAck, Busy, nFault, PError, Select, or PLH) change state.

Disconnect Interrupt

The Disconnect interrupt is enabled by setting the Disconnect Interrupt Mask bit in the USS-720 Control Register to 0. There is no Interrupt Status bit associated with the Disconnect interrupt. This interrupt is generated when the Peripheral Logic High signal makes a transition from 1 to 0, or when Peripheral Logic High is 0 and all other parallel port signal lines driven by the peripheral (nAck, Busy, nFault, PError, and Select) are high for longer than one second. Either of these conditions should indicate that the peripheral has been disconnected from the USS-720.

External Circuitry Requirements

The USS-720 is intended to be a single-chip solution. As such, the USB transceiver and the *IEEE* 1284 buffers have been integrated on-chip. External requirements include a 3.3 V supply and a 1.5 kΩ ± 5% pull-up resistor for the DPLS pin. If the internal oscillator is used, a 12 MHz crystal along with bias capacitors are needed (see Figure 3). If the internal oscillator is not used, a 12 MHz clock signal should be supplied to CLK_LO, and CLK_HI should be left unconnected. 1.2 kΩ ± 5% pull-up resistors are needed on all *IEEE* 1284 signals (except PLH), unless used in the High Drive Mode (see the following page). A 5 V supply and USB and/or *IEEE* 1284 connectors might also be needed, depending on the application.

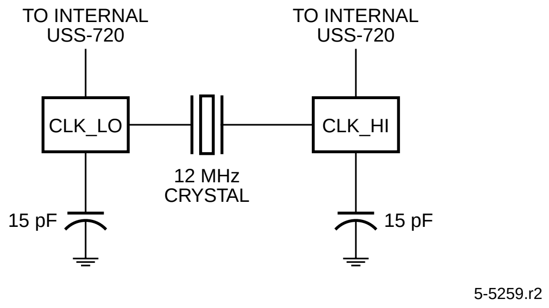


Figure 3. External Crystal Connection

Figure 4 shows the normal USS-720 connection to the USB. Both DPLS and DMNS require 24 Ω ± 1% series resistors for USB impedance matching. Additionally, a 1.5 kΩ pull-up resistor is required on DPLS for full-speed/low-speed differentiation.

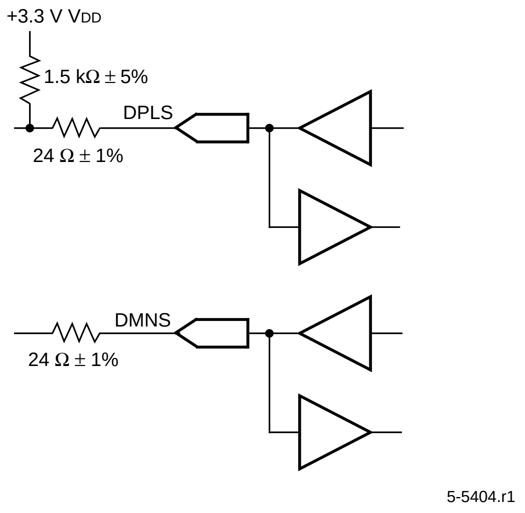


Figure 4. USB Transceiver Connection (Normal Mode)

For using the USS-720 in a self-powered device, there are some additional considerations. The device must refrain from supplying power through the pull-up resistor when the device is plugged into an unpowered bus. The USS-720 device circuit must also ensure that the DPLS and DMNS lines are in an appropriate state when the device is powered but not plugged in. Figure 5 shows an example connection to meet these requirements.

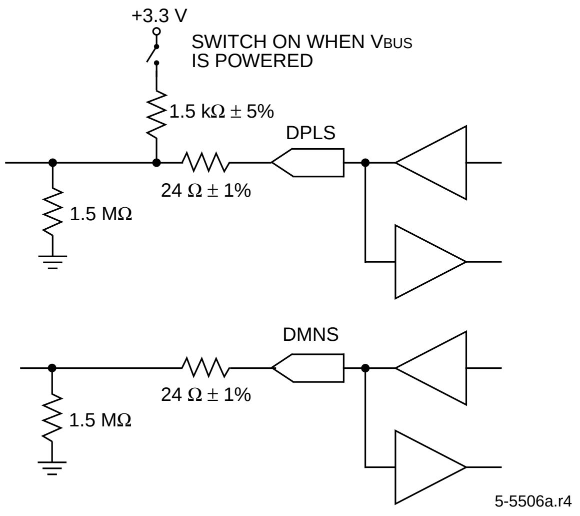


Figure 5. Self-Powered Device Example Connection

Figure 6 shows a USS-720-to-*IEEE* 1284 parallel port connection that complies with the *IEEE* 1284 specification. Other connections are also possible. While the *IEEE* specification requires these resistors, developers must make their own design decision against the 500 µA suspend mode current requirements required by the USB specification.

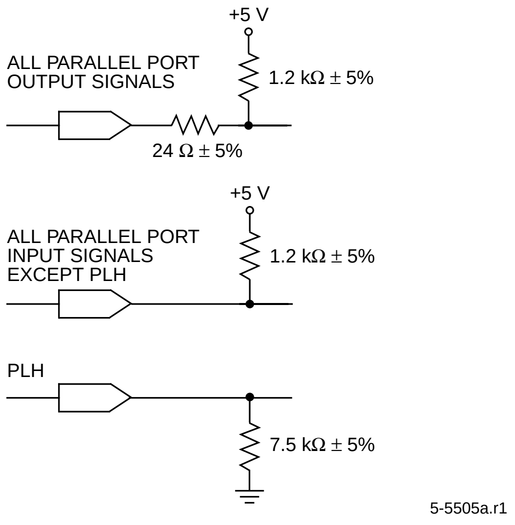

+5 V
ALL PARALLEL PORT OUTPUT SIGNALS
1.2 kΩ ± 5%
24 Ω ± 5%
+5 V
ALL PARALLEL PORT INPUT SIGNALS EXCEPT PLH
1.2 kΩ ± 5%
PLH
7.5 kΩ ± 5%
5-5505a.r1

Figure 6. USS-720 Connection to *IEEE* 1284

Filter Bypass Mode

For embedded applications, the USS-720 *IEEE* 1284 port can be operated in Filter Bypass Mode. This mode disables digital filtering of the parallel port signals into the USS-720, providing a performance improvement.

Note: Since digital filtering is disabled, the parallel port lines will be susceptible to noise. Do **not** use this mode when driving across a cable.

Filter Bypass Mode is enabled by connecting a 20 kΩ pull-down resistor to the SUSPEND line.

High Drive Mode

If desired (e.g., for embedded applications), the USS-720 *IEEE* 1284 port can operate in High Drive Mode. In this mode, all parallel port signals are constantly driven by 14 mA totem-pole drivers, rather than with the normal open-drain drivers. This eliminates the need for external pull-up resistors on the parallel port signals (and, if driving another chip on the same board, there is no need for the 24 Ω impedance matching resistors). High Drive Mode is enabled by connecting a 20 kΩ pull-down resistor to the SK line.

Self-Powered Mode

When using the USS-720 in a self-powered application, attach a 5 kΩ pull-up resistor to the CS line. This causes the correct self-powered status to be reported in response to a USB Get-Status command.

Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operations sections of this data sheet. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

Table 30. Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Ambient Operating Temperature Range	T_A	0	70	°C
Storage Temperature	T_{stg}	–40	125	°C
Voltage on Any Pin with Respect to Ground	V_{IN}	V_{SS} – 0.3	V_{DD} + 0.3*	V

* Except for 5 V tolerant buffers where V_{IN} max = V_{DD5} max + 0.3 V. V_{IN} must never exceed V_{DD5} + 0.3 V at any time. V_{DD5} should be selected to satisfy this condition.

Electrical Characteristics

Table 31. dc Characteristics (T_A = 0 °C to 70 °C, V_{DD} = 3.3 V ± 0.3 V, V_{SS} = 0 V.)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Input Voltage:						
Low	V_{IL}	—	—	—	0.8	V
High	V_{IH}	—	2.0	—	—	V
Output Voltage:						
Low	V_{OL}	—	—	—	0.4	V
High	V_{OH}	—	2.4	—	—	V
Power Dissipation	P_D	25 °C, V_{DD} = 3.3 V	1.65	231	323.4	mW
Power Supply Voltage	V_{DD}, V_{DDA}	—	3	3.3	3.6	V
	V_{DD5}	5 V environment	4.375	5	5.5	V
		3 V environment	3	3.3	3.6	V
Power Supply Current	I_{DD}	—	0.5	70	98	mA

Table 32. USB Transceiver Driver Characteristics

Parameter	Symbol	Test Conditions	Min	Max	Unit
Rise and Fall Times:		OEN = 0, C_L = 50 pF			
(10%—90%)	t_R		4	20	ns
(90%—10%)	t_F		4	20	ns
Rise/Fall Time Matching	t_{RFM}	OEN = 0, C_L = 50 pF	90	110	%
Crossover Point	V_{CRS}	OEN = 0, C_L = 50 pF	1.3	2.0	V
Output Impedance*	Z_{DRV}	OEN = 0	28	43	Ω

* The output impedance includes both the external resistor and the transceiver.

The USS-720 is a 3.3 V part, and it has separate pins (V_{DD5}) for power to the *IEEE* 1284 drivers. Capacitance values for the USS-720 pins are listed in Table 33.

Table 33. Capacitance Values

Parameter	Value	Unit
CLK_LO	1.0	pF
CLK_HI	1.0	pF
All Other Pins	3.0	pF

Timing Characteristics

- Timing is specified over the operating range from 0 °C to 70 °C ambient temperature, V_{DD} = 3.0 V to 3.6 V, and $V_{DD}5$ = 4.75 V to 5.25 V.
- All timing is referenced from the rising edge of CLK_LO, with 70 pF output load.
- Only DIO is required to meet these input setup and hold times for proper operation. For other inputs, meeting the limit is only required to ensure that the signal will be recognized on the referenced clock edge rather than one period later.

Table 34. Setup and Hold Input Timing

Parameter	Setup Time	Hold Time	Unit
DIO Setup and Hold	2.9	3.4	ns

Table 35. Clock Characteristics/Miscellaneous Input Timing

Parameter	Symbol	Min	Typical	Max	Unit
Frequency of Operation (CLK_LO)	f	11.976	12.000	12.024	MHz
Clock Period	tcyc	83.1	83.3	83.5	ns
Duty Cycle	—	—	40/60*	—	%

* The duty cycle applies to any frequency in the specified range.

Table 36. Output Delay Timing

Parameter	Min	Max	Unit
PDATA Output Delay	5.3	21.5	ns
NUSB_RESET Output Delay	6.5	17.1	ns
SUSPEND Output Delay	2.6	34.6	ns
SK Output Delay	6.4	16.0	ns
CS Output Delay	—	75.0	ns
DIO Output Delay	3.6	31.6	ns

Table 37. Miscellaneous Output Timing

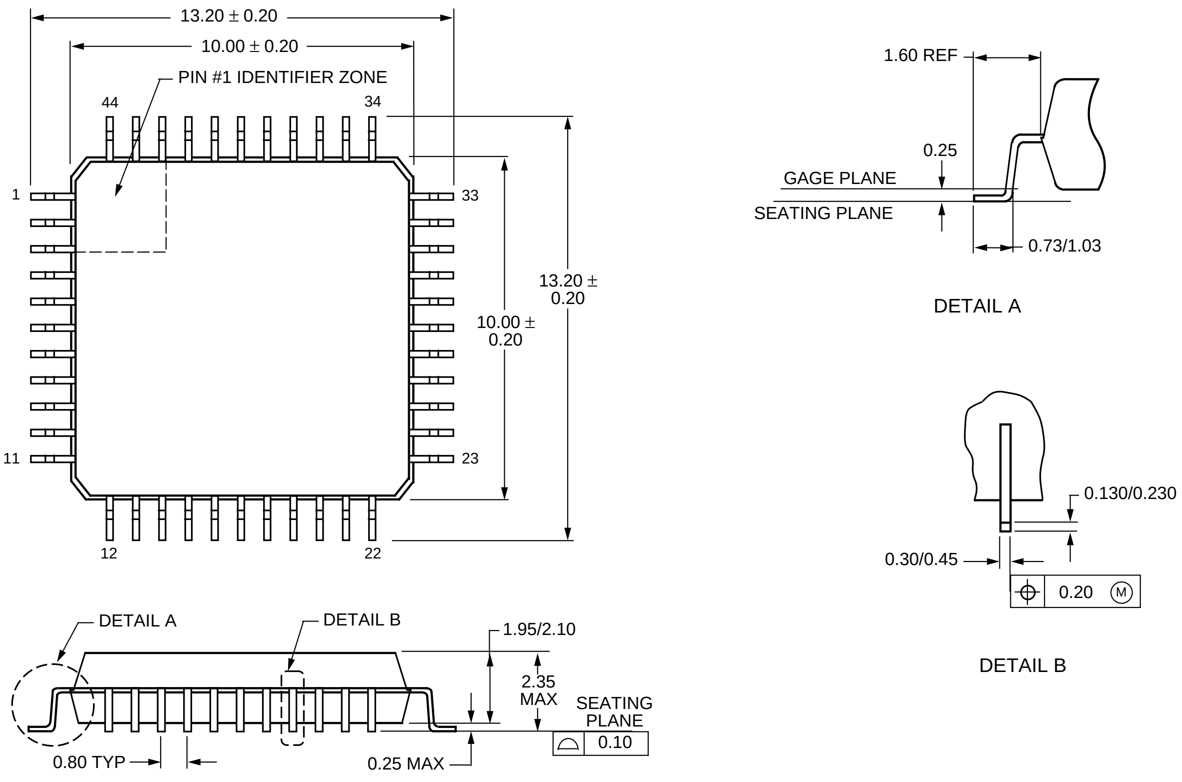
Parameter	Symbol	Min	Typical	Max	Unit
Frequency of Operation (SK)	f	390	429	476	kHz
Clock Period	tcyc	2.10	2.33	2.56	µs
Duty Cycle	—	—	50/50*	—	%

* The duty cycle applies to any frequency in the specified range.

Outline Diagram

44-Pin MQFP

Dimensions are in millimeters.



5-2111.r11

Ordering Information

Device Code	Package	Comcode
USS720E-DB	44-Pin MQFP	108127085

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

Typical Circuit Showing the USS-720 Bridging USB to Parallel Port

Description

The USS-720 can be used in a variety of applications, such as bus-powered devices, self-powered devices, hubs, and embedded printer controllers. The following describes using the device in a bus-powered application (see attached schematic).

Note: The attached schematic depicts a typical functional circuit using the USS-720 as a bridge between the USB and a parallel port. Actual applications may require additional protection circuitry. The schematic and circuit description provided in this application note are for reference purposes only. Neither Lucent nor In-System Design warrants their suitability for any particular purpose.

The USS-720 is a dual-powered chip requiring both 5 V and 3.3 V supplies. The 3.3 V is generated using a low dropout regulator. The USS-720 must operate with a USB supply (VBUS) of 4.4 V to 5.25 V. Using a low dropout regulator ensures a solid 3.3 V supply even at the lowest limits of the 5 V VBUS. The 5 V supply is used by the 1284 printer port drivers.

The USS-720 also requires a 12 MHz ± 0.25% crystal. In embedded applications, an oscillator output should be connected to clk_lo (pin 21), and pin 22 should be left unconnected.

The USS-720 requires a 1.5 kΩ pull-up resistor attached to the DPLS signal to indicate that it is a high-speed 12 Mbits/s device as per the USB specification. There is also a USB differential driver impedance specification of 30 Ω to 42 Ω. A 24 Ω series resistor, when added to the output impedance of the USS-720 USB drivers, puts the total output impedance in the middle of that range.

A similar driver output impedance requirement is true for the *IEEE** 1284 printer signals. A 24 Ω series resistor is used on all the 1284 signals that are driven by the USS-720 to give each signal a total output impedance of 45 Ω to 55 Ω . The *IEEE* 1284 specification also requires 1.2 kΩ pull-up resistors on all the printer signals, except PLH. PLH only requires a 7.5 kΩ pull-down resistor.

While the *IEEE* 1284 specification requires these resistors, developers must make their own design decisions against the 500 µA suspend mode current requirements of the USB specification. The following schematic does not show the pull-down resistor on the PLH signal. The internal pull-down on the USS-720 can be used for this purpose. Similarly, the pull-up resistor values may have to be modified to higher values than the *IEEE* 1284 specification allows in order to meet the USB requirements.

Standard decoupling should be used on the board. It is recommended that 0.1 µF capacitors are placed between VCC_5V/VCC and GND and that they are located as close as possible to the power pins on the USS-720. Sufficient grounding must be implemented on the board to ensure proper functionality. A four-layer board design is recommended with two of the layers used for power and ground planes.

The test and scan signals (pins 23, 24, 31, 32, and 33) are used for functional testing after fabrication and are tied low during normal operation. It is recommended that typical bypass techniques be used on the voltage supply pins (see ASIC BYPASS CAPS on attached schematic).

The USS-720 contains a small amount of ROM space that is used to store descriptor data. This data is used during the Plug and Play mode in a *Microsoft Windows*† operating system to identify the product. This onboard ROM data can be used during the initial development phase, but unique descriptor data must be provided by the USB peripheral developer for each design.

* *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.
† *Microsoft* and *Windows* are registered trademarks of Microsoft Corporation.

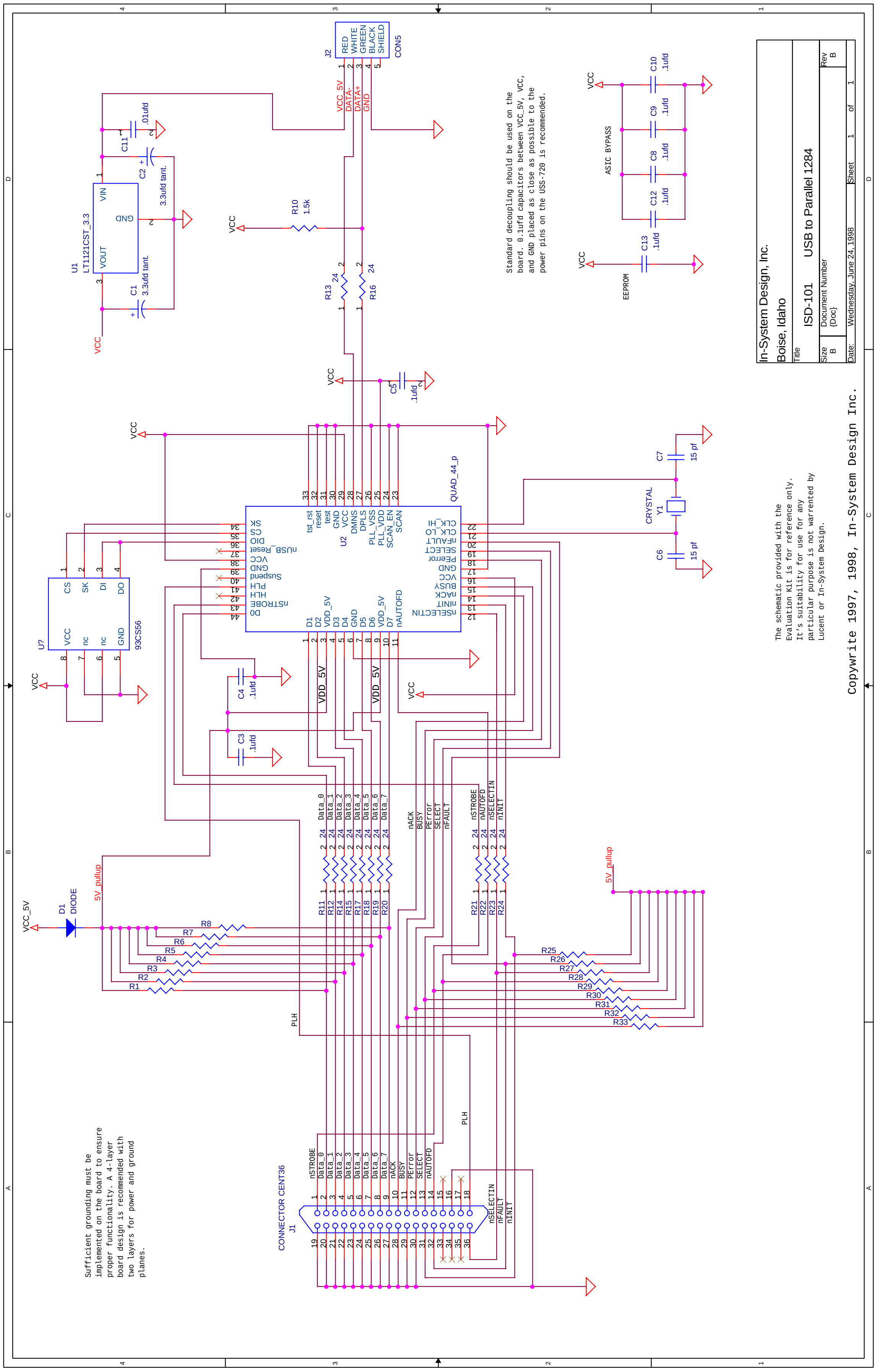
Description (continued)

Whenever device descriptor data is requested, the USS-720 drives both control pins CS (pin 35, Serial ROM Chip Select) and SK (pin 34, Serial ROM Clock). The USS-720 then looks for a response on DIO (pin 36, Serial ROM Data Signal). If there is no external device connected, and no data is present on the DIO pin, then the descriptor data is taken from the internal ROM. The USS-720 accesses this data through a *MicroWire** ROM Interface. One method for this is to use an EEPROM such as the 93CS56L/66L. The USS-720 Evaluation Kit is specifically designed for use with this part. Substitution components must be pin and functional compatible with the 93CS56L/66L. 93C56L/66L, 93CS46L, and 93C46L EEPROM parts will **not** work. The connection scheme for this device is shown on the attached schematic. The circuit diagram assumes that a preprogrammed EEPROM is used.

Note: Developers using the USS-720 must use an external serial EEPROM (or the equivalent) in their design and create their own hex data file for use in programming the EEPROM at their site. The configuration data stored in this serial EEPROM is used by the *Microsoft* host software during the enumeration to load the appropriate drivers. Using unique identification data in the EEPROM provides a means for the developer to ensure that only their software is loaded for use with their device. Otherwise, the enumeration of a camera using the USS-720 could cause the software for a USS-720-based printer to be loaded, resulting in a system that does not function correctly.

See the Device Descriptor, Configurations, and Interfaces section in *USS-720 USB-to-*IEEE *1284 Bridge* Data Sheet for more information on device descriptors.

* *MicroWire* is a registered trademark of Advanced Interconnection Technology, Inc.



Standard decoupling should be used on the board. 0.1ufd capacitors between VCC_5V, VCC, and GND placed as close as possible to the power pins on the USS-720 is recommended.

The schematic provided with the Evaluation Kit is for reference only. It's suitability for use for any particular purpose is not warranted by Lucent or In-System Design.

Copywrite 1997, 1998, In-System Design Inc.

Sufficient grounding must be implemented on the board to ensure proper functionality. A 4-layer board design is recommended with two layers for power and ground planes.

Figure 1. USS-720 Evaluation Kit Schematic

Preliminary User Guide, Rev. 2
February 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

USS-720 USB Device Driver

1 Introduction

This document describes the design and implementation of USS720.SYS, a *Windows** WDM USB driver developed by In-System Design, Inc. (Note that this driver was written specifically to work with the In-System Design USB Smart Cable. For vendor-specific implementations, refer to the “Notes for Developers” in the Introduction of this Information Manual.) The purpose of USS720.SYS is to provide bulk data transfers and a mapping layer for the *IEEE*† 1284 registers over the Universal Serial Bus interface.

The purpose of this document is to:

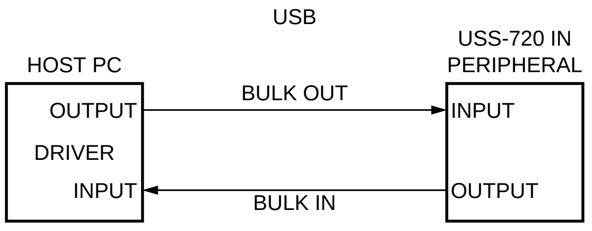
- Define how USS720.SYS will interact with the *Windows* operating system components and the interface layer to other device classes.
- Provide a guide for using USS720.SYS.

Note: The USS-720’s Automatic Mode is recommended for almost all applications. The drivers included in the Lucent Technologies USS-720 Evaluation Kit use only Automatic Mode. Automatic mode implements all negotiation handshakes automatically for Compatibility, Nibble, and ECP modes. In Register Mode, the user must do the negotiations manually in software. Register Mode can be useful when implementing a nonstandard parallel port interface.

For more information on the *Windows* system-level components, USB architecture, and 1284 hardware, please refer to:

- *Microsoft Win32 SDK* (October 96 or later), available from *Microsoft* *.
- *WDM Driver Model Specification*, available from *Microsoft.*
- *WDM USB Driver Specification*, available from *Microsoft.*
- *USB Device Class Definition for Printing Devices,* from usb.org web site.
- *USS-720 USB-to-*IEEE *1284 Bridge* Preliminary Data Sheet (DS98-393CMPR), available from Lucent Technologies Microelectronics Group.
- IEEE *Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers* (*IEEE* Standard 1284-1994)

Figure 1 provides a point of reference for directions mentioned in this document.



5-6005.r2

Figure 1. Signal Directions

* *Windows* and *Microsoft* are registered trademarks of Microsoft Corporation.
† *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.

Table of Contents

2 Initialization

- The driver entry routine is called and initializes callbacks for the following functions when the operating system loads the driver:

— Create:	Open a handle to a device instance.
— Read:	Read a block of data from the Bulk In pipe.
— Write:	Write a block of data to the Bulk Out pipe.
— DeviceIoControl:	Process I/O requests documented in Section 4 of this document.
— Close:	Close the handle of the device instance.

- A symbolic link will be created associating this device driver with a special device instance that can be used to call IOCTL_GET_DEVICE_INSTANCES (i.e., LPTUSB). (See Section 4.13 of this document.)
- The Add_Device routine is called by the configuration manager. The configuration manager interacts with the USB bus enumerator to identify newly attached devices in the system. Through the layered driver architecture with the host controller driver, this USS-720 routine will be called when a USS-720 device is identified. After the device object has been successfully created, symbolic links will be created associating this device driver with the specific device instances (i.e., LPTUSB1, LPTUSB2, LPTUSBx).

3 I/O File Functions

This section provides instructions on how to use the I/O file functions to interface with the USS-720. Only parameters relative to the USS-720 Device Driver are explained in this document. Please refer to the *Win32 SDK* documentation for the complete definitions of the following functions and all referenced data types:

Function	Description
CreateFile	Open an instance of a particular device.
ReadFile	Reads data from the Bulk In pipe.
WriteFile	Write data to the Bulk Out pipe.
DeviceIoControl	Send a control code to a device instance.
CloseHandle	Close an instance of a device.

3 I/O File Functions (continued)

3.1 CreateFile

The CreateFile I/O file function opens an instance of a particular device and returns an open handle to the device instance. Details regarding the syntax, parameters, return values, and error codes are listed below.

3.1.1 Syntax

```
HANDLE
CreateFile(
    LPCTSTR                  lpDevName,
    DWORD                    dwDesiredAccess,
    DWORD                    dwShareMode,
    LPSECURITY_ATTRIBUTES    lpSecurityAttributes,
    DWORD                    dwCreationDistribution,
    DWORD                    dwFlagsAndAttributes,
    HANDLE                   hTemplateFile
    );
```

3.1.2 Parameters

Input:
lpDevName—Points to a null-terminated string that specifies the symbolic link name of a specific device instance to open (see Section 4.12 IOCTL_GET_DEVICE_INSTANCES). Note that the symbolic link must be prefixed with “\\.\”. For example, to retrieve the handle of the symbolic name LPTUSB1, “\\.\LPTUSB1” must be passed to the lpDevName parameter.

Note: For definitions of other input parameters, refer to CreateFile Function in the *WIN32 SDK* documentation.

Output:
None.

3.1.3 Return Values

If the function succeeds, the return value is an open handle to the device instance.

If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call GetLastError.

3.1.4 Notes

See the *Win32 SDK* documentation for complete definitions of parameters.

3.1.5 Error Code

ERROR_ALREADY_OPENED—Device instance is already open.

3 I/O File Functions (continued)

3.2 ReadFile

The ReadFile I/O file function reads data from the Bulk In pipe and returns TRUE if the function is successful and FALSE if it fails. Details regarding the syntax, parameters, return values, and error codes are listed below.

3.2.1 Syntax

```
BOOL
ReadFile(
    HANDLE              hDev,
    LPVOID              lpBuffer,
    DWORD               nNumberOfBytesToRead,
    LPDWORD             lpNumberOfBytesRead,
    LPOVERLAPPED        lpOverlapped
    );
```

3.2.2 Parameters

Input:
hDev—Handle returned from a successful CreateFile.

nNumberOfBytesToRead—Specifies the number of bytes to read from the device.

lpOverlapped—Pointer to an OVERLAPPED structure. This is an optional parameter that can be “NULL” if overlapping I/O is not required.

Output:
lpBuffer—Pointer to the buffer to receive data from the Bulk In pipe of the device. This buffer is assumed to be in locked system memory allocated from the nonpaged pool.

lpNumberOfBytesRead—Pointer to the number of bytes that were read.

3.2.3 Return Values

If the function succeeds, the return value is TRUE (1).

If the function fails, the return value is FALSE (0). To get extended error information, call GetLastError.

3.2.4 Notes

See the *Win32 SDK* documentation for complete definitions.

3.2.5 Error Codes

ERROR_DEVICE_REMOVE—The device instance has been removed.

ERROR_DEVICE_STOP—The device instance has been stopped.

ERROR_INVALID_ALTERNATESETTING—The current alternate interface setting for this device instance does not support the Bulk In pipe.

ERROR_PIPE_ALREADY_OPEN—The Bulk In pipe for this device instance is already open.

STATUS_INVALID_HANDLE—The Bulk In pipe handle is invalid for this device instance.

STATUS_INVALID_PARAMETER—The pipe type is invalid for this device instance.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to allocate a request to read Bulk In on this device instance.

3 I/O File Functions (continued)

3.3 WriteFile

The WriteFile I/O file function writes data to the Bulk Out pipe and returns TRUE if the function is successful and FALSE if it fails. Details regarding the syntax, parameters, return values, and error codes are listed below.

3.3.1 Syntax

```
BOOL
WriteFile(
    HANDLE                  hDev,
    LPVOID                  lpBuffer,
    DWORD                   nNumberOfBytesToWrite,
    LPDWORD                 lpNumberOfBytesWritten,
    LPOVERLAPPED            lpOverlapped
    );
```

3.3.2 Parameters

Input:
hDev—Handle returned from a successful CreateFile.

nNumberOfBytesToWrite—Specifies the number of bytes to be written.

lpOverlapped—Pointer to OVERLAPPED structure. This is an optional parameter that can be “NULL” if overlapping I/O is not required.

Output:
lpBuffer—Pointer to the buffer containing the data to be written to the Bulk Out pipe of the device. This buffer is assumed to be in locked system memory allocated from the nonpaged pool.

lpNumberOfBytesWritten—Pointer to the number of bytes written.

3.3.3 Return Values

If the function succeeds, the return value is TRUE (1).

If the function fails, the return value is FALSE (0). To get extended error information, call GetLastError.

3.3.4 Notes

See the *Win32 SDK* documentation for complete definitions.

3.3.5 Error Codes

ERROR_DEVICE_REMOVE—The device instance has been removed.

ERROR_DEVICE_STOP—The device instance has been stopped.

ERROR_INVALID_ALTERNATESETTING—The current alternate interface setting for this device instance does not support the Bulk Out pipe.

ERROR_PIPE_ALREADY_OPEN—The Bulk Out pipe for this device instance is already open.

STATUS_INVALID_HANDLE—The Bulk Out pipe handle is invalid for this device instance.

STATUS_INVALID_PARAMETER—The pipe type is invalid for this device instance.

STATUS_INSUFFICIENT_RESOURCES—This is an internal error resulting from insufficient memory available to perform a required USB request.

3 I/O File Functions (continued)

3.4 DeviceIoControl

The DeviceIoControl I/O file function sends a control code to a device instance and returns TRUE if the function is successful and FALSE if it fails. Details regarding the syntax, parameters, return values, and error codes are listed below.

3.4.1 Syntax

```
    BOOL
DeviceIoControl(
    HANDLE                  hDev,
    DWORD                   dwIoControlCode,
    LPVOID                  lpInBuffer,
    DWORD                   nInBufferSize,
    LPVOID                  lpOutBuffer,
    DWORD                   nOutBufferSize,
    LPDWORD                 lpBytesReturned,
    LPOVERLAPPED            lpOverlapped
    );
```

3.4.2 Parameters

Input:
hDev—Handle returned from a successful CreateFile.

dwIoControlCode—Specifies the control code for the operation. See Section 4 of this document for a list of valid I/O control codes and more detailed information on each control code.

lpInBuffer—Pointer to a buffer that contains the data required to perform the operation. This parameter can be NULL if the dwIoControlCode parameter specifies an operation that does not require input data. The usage of lpIn-Buffer is dependent on the dwIoControlCode parameter.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. The usage of nInBufferSize is dependent on dwIoControlCode parameter.

lpOverlapped—Pointer to OVERLAPPED structure. This is an optional parameter that can be “NULL” if overlapping I/O is not required.

Output:
lpOutBufferSize—Pointer to a buffer that receives the operation’s output data. This parameter can be NULL if the dwIoControlCode parameter specifies an operation that does not require output data. The usage of lpOutBufferSize is dependent on dwIoControlCode parameter.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. The usage of nOutBufferSize is dependent on dwIoControlCode parameter.

lpBytesReturned—Pointer to a variable that receives the size (in bytes) of data stored into the output buffer pointed to by lpOutBuffer. The usage of lpBytesReturned is dependent on dwIoControlCode parameter.

3.4.3 Return Values

If the function succeeds, the return value is TRUE (1).

If the function fails, the return value is FALSE (0). To get extended error information, call GetLastError.

3.4.4 Notes

See the *Win32 SDK* documentation for complete definitions.

3 I/O File Functions (continued)

3.4.5 Error Codes

ERROR_DEVICE_REMOVE—The device instance has been removed.

ERROR_DEVICE_STOP—The device instance has been stopped.

STATUS_INVALID_PARAMETER—Control Code is either invalid for this device instance or the device instance is in a state that cannot process the Control Code.

See specific Control Code for other errors (see Section 4 of this document).

3.5 CloseHandle

The CloseHandle I/O function closes an instance of a device and returns TRUE if the function is successful and FALSE if it fails. Details regarding the syntax, parameters, return values, and error codes are listed below.

3.5.1 Syntax

```
BOOL
CloseHandle(
    HANDLE hDev
    );
```

3.5.2 Parameters

Input:
hDev—Handle returned from a successful CreateFile.

Output:
None.

3.5.3 Return Values

If the function succeeds, the return value is TRUE (1).

If the function fails, the return value is FALSE (0). To get extended error information, call GetLastError.

3.5.4 Notes

See the *Win32 SDK* documentation for complete definitions.

3.5.5 Error Codes

See *WIN32 SDK* documentation for possible errors.

4 I/O Control Codes

This section provides information for the I/O control codes used in the DeviceIoControl file function (see Section 3.4). These control codes are used in the DeviceIoControl function to perform any of the following device-specific operations.

Note: Refer to the header file <USS720io.h> included on the USS-720 Evaluation Kit diskette for details regarding these control codes definitions. Please note that this file may be subject to periodic modifications. To obtain more information about these control codes, contact In-System Design.

Control Code	Description
IOCTL_1284_ ECP_FWDTOREV	Negotiates the peripheral from forward idle to reverse while in ECP register mode.
IOCTL_1284_ECP_REVTOFWD	Negotiates the peripheral from reverse to forward idle while in ECP register mode.
IOCTL_1284_ ECP_SET_CHANNEL	Sets the ECP channel on the peripheral for reads and writes.
IOCTL_1284_ SET_MODE	Negotiates the peripheral into one of the valid register modes.
IOCTL_1284_ TERMINATE	Performs a standard 1284 termination sequence.
IOCTL_ABORT_PIPE	Cancels any pending transfers for the specified pipe. The pipe state and endpoint state are unaffected.
IOCTL_CANCEL_PIPE_REQUEST	Cancels the current request on the specified pipe by flushing the pipe and canceling any outstanding requests on that pipe.
IOCTL_GET_ALTSETTING	Retrieves the current alternate interface setting from the USS-720 device.
IOCTL _GET_CAPABILITIES	Returns a variable buffer length containing the device capabilities string. This is a *IEEE* 1284 compatible string.
IOCTL_GET_CONFIGURATION_DESCRIPTOR	Retrieves the current configuration descriptor.
IOCTL_GET_DEVICE_DESCRIPTOR	Receives the current device descriptor.
IOCTL_GET_DEVICE_INSTANCES	Returns a list of current devices based on the symbolic links created during each device enumeration.
IOCTL_GET_INTERFACE	Gets the information about the current interface and pipes.
IOCTL _GET_PORT_STATUS	Returns a status byte.
IOCTL_ISSUE_USS720_COMMAND	Issues a specific command for the USS-720 device.
IOCTL _GET_1284_REGISTER	Returns all 1284 registers.
IOCTL_READ_INTERRUPT_PIPE	Reports changes in the parallel port and buffer status when they occur.
IOCTL_RESET_PIPE	Clears the halted state of the specified pipe within the USB stack and resets the stalled state of the endpoint on the device.
IOCTL _SET_1284_REGISTER	Sets one of the 1284 registers.
IOCTL _SET_ALTSETTING	Sets the alternate interface setting.
IOCTL _SOFT_RESET	Resets the device, flushes the Bulk Out and Bulk In pipes to the default states.

4 I/O Control Codes (continued)

4.1 IOCTL_1284_ ECP_FWDTOREV

IOCTL_1284_ECP_FWDTOREV negotiates the peripheral from forward idle to reverse while in ECP register mode. Details regarding parameters and error codes follow.

4.1.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.1.2 Notes

Requires that the register-based mode be successfully set to ECP_REGISTER_MODE or ECP_RLE_REGISTER_MODE (see *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operation” and Section 4.4, IOCTL_1284_SET_MODE, of this document).

4.1.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

STATUS_IO_DEVICE_ERROR—Negotiation failed.

4 I/O Control Codes (continued)

4.2 IOCTL_1284_ECP_REVTOFWD

IOCTL_1284_ECP_REVTOFWD negotiates the peripheral from reverse to forward idle while in ECP register mode. Details regarding parameters and error codes follow.

4.2.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.2.2 Notes

Requires that the register-based mode be successfully set to ECP_REGISTER_MODE or ECP_RLE_REGISTER_MODE (see *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operation” and Section 4.4, IOCTL_1284_SET_MODE, of this document).

4.2.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9).

STATUS_IO_DEVICE_ERROR—Negotiation failed.

4 I/O Control Codes (continued)

4.3 IOCTL_1284_ ECP_SET_CHANNEL

IOCTL_1284_ ECP_SET_CHANNEL sets the ECP channel on the peripheral for reads and writes. Details regarding parameters and error codes follow.

4.3.1 Parameters

lpInBuffer—Points to a buffer that contains a single byte which specifies the ECP channel to be set.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.3.2 Notes

Requires that the register-based mode be successfully set to ECP_REGISTER_MODE or ECP_RLE_REGISTER_MODE (see *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operation” and Section 4.4, IOCTL_1284_SET_MODE, of this document).

4.3.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

STATUS_IO_DEVICE_ERROR—Negotiation failed.

4 I/O Control Codes (continued)

4.4 IOCTL_1284_ SET_MODE

IOCTL_1284_ SET_MODE negotiates the peripheral into one of the valid register modes. Details regarding parameters and error codes follow.

4.4.1 Parameters

lpInBuffer—Points to a buffer that contains a single byte that specifies one of the following register modes:

Register Mode	Description
STANDARD_REGISTER_MODE	Sets USS-720 into register-based Standard Mode.
BIDIRECTIONAL_REGISTER_MODE	Sets USS-720 into register-based Bidirectional Mode.
COMPATIBILITY_REGISTER_MODE	Sets USS-720 into register-based Compatibility Mode.
ECP_REGISTER_MODE	Negotiates the USS-720 into register-based ECP Mode without RLE.
ECP_RLE_REGISTER_MODE	Negotiates the USS-720 into register-based ECP Mode with RLE.
EPP_REGISTER_MODE	Currently not supported.*

* Using EPP mode is possible in the USS-720 for users capable of writing their own software based on the USS-720 data sheet information. However, the USS720.SYS driver does not support EPP mode applications.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.4.2 Notes

All operations expect the USS-720 device to have been reset (see IOCTL_SOFT_RESET) before executing this command.

In standard, bidirectional, and compatibility register modes, no 1284 negotiation takes place. The USS-720 device is simply placed into the appropriate register-based operation. (See *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operations.”)

In ECP_REGISTER_MODE and ECP_RLE_REGISTER_MODE, the USS-720 device is placed into ECP register-based operation after correctly negotiating into ECP mode via the Control and Status registers. (See *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operations.”)

4.4.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

ERROR_INVALID_REGISTER_MODE—Register mode is either invalid or not currently supported for the device instance.

STATUS_IO_DEVICE_ERROR—Negotiation failed.

4 I/O Control Codes (continued)

4.5 IOCTL_1284_ TERMINATE

IOCTL_1284_ TERMINATE performs a standard 1284 termination sequence. Details regarding parameters and error codes follow.

4.5.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.5.2 Notes

Requires that the register-based mode be successfully set to ECP_REGISTER_MODE or ECP_RLE_REGISTER_MODE (see *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Register-Based Operation” and Section 4.4, IOCTL_1284_SET_MODE, of this document).

4.5.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9).

STATUS_IO_DEVICE_ERROR—Standard termination failed.

4 I/O Control Codes (continued)

4.6 IOCTL_ABORT_PIPE

IOCTL_ABORT_PIPE cancels any pending transfers for the specified pipe. The pipe state and endpoint state are unaffected. Details regarding parameters and error codes follow.

4.6.1 Parameters

lpInBuffer—Points to a buffer that contains a single byte that specifies one of the following values:

Value	Description
BULK_OUT_PIPE	Bulk Out pipe.
BULK_IN_PIPE	Bulk In pipe.
INTERRUPT_PIPE	Interrupt pipe.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.6.2 Notes

None.

4.6.3 Error Codes

ERROR_INVALID_PIPE_INDEX—Specified pipe is invalid for this device instance or invalid for the current alternate interface setting.

STATUS_INVALID_PARAMETER—nInBufferSize cannot be zero.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to allocate a request to abort the pipe on this device instance.

4 I/O Control Codes (continued)

4.7 IOCTL_CANCEL_PIPE_REQUEST

IOCTL_CANCEL_PIPE_REQUEST cancels the current request on the specified pipe by flushing the pipe and canceling any outstanding requests on the pipe. Details regarding parameters and error codes follow.

4.7.1 Parameters

lpInBuffer—Points to a buffer that contains a single byte that specifies one of the following values:

Value	Description
BULK_OUT_PIPE	Bulk Out pipe.
BULK_IN_PIPE	Bulk In pipe.
INTERRUPT_PIPE	Interrupt pipe.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.7.2 Notes

None.

4.7.3 Error Codes

ERROR_INVALID_PIPE_INDEX—Specified pipe is invalid for this device instance or invalid for the current alternate interface setting.

STATUS_INVALID_PARAMETER—nInBufferSize cannot be zero.

4 I/O Control Codes (continued)

4.8 IOCTL_GET_1284_REGISTER

IOCTL_GET_1284_REGISTER returns all 1284 registers. Details regarding parameters and error codes follow.

4.8.1 Parameters

lpInBuffer—Points to an input buffer that contains a byte that specifies the address of the parallel port register to be read.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. This buffer must be large enough to contain one byte value.

lpOutBuffer—Points to an output buffer that will receive a REGISTER_1284 structure or an ADVREGISTER_1284 structure (see Sections 5.3 and 5.4 of this document).

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer, which must be large enough to contain a REGISTER_1284 data structure or an ADVREGISTER_1284 structure. The size specified in nOutBuffer-Size will determine whether REGISTER_1284 or ADVREGISTER_1284 is returned (see Sections 5.3 and 5.4 of this document).

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.8.2 Notes

See the *USS-720 USB-to-*IEEE *1284 Bridge* Preliminary Data Sheet, “GET_1284_REGISTER” for more details on the vendor-specific requests.

4.8.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to receive a REGISTER_1284 structure (see Section 5.3 of this document).

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.9 IOCTL_GET_ALTSETTING

IOCTL_GET_ALTSETTING retrieves the current alternate interface setting from the USS-720 device. Details regarding parameters and error codes follow.

4.9.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive a single byte indicating the current alternate setting based on one of the following:

Value	Description
ALT_INTERFACE_0	Bulk Out pipe only.
ALT_INTERFACE_1	Bulk Out and Bulk In pipe.
ALT_INTERFACE_2	Bulk Out, Bulk In, Interrupt pipe, and 1284 register capabilities.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. The buffer must be large enough to contain one byte value.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.9.2 Notes

None.

4.9.3 Error Code

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to contain one byte value.

4 I/O Control Codes (continued)

4.10 IOCTL_GET_CAPABILITIES

IOCTL_GET_CAPABILITIES returns a variable buffer length containing the device capabilities string. This is an *IEEE* 1284 compatible string. Details regarding parameters and error codes follow.

4.10.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive an *IEEE* 1284 Device ID string.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Specify zero to return only the length of the Device ID string in lpBytesReturned.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer or returns the length of the Device ID string if nOutBufferSize is zero.

4.10.2 Notes

None.

4.10.3 Error Codes

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer was not large enough to receive the Device ID string (see *USS-720 USB-to-*IEEE *1284 Bridge* Preliminary Data Sheet, “Printer Class-Specific Requests”).

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.11 IOCTL_GET_CONFIGURATION_DESCRIPTOR

IOCTL_GET_CONFIGURATION_DESCRIPTOR retrieves the current configuration descriptor. Details regarding parameters and error codes follow.

4.11.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive a USB_CONFIGURATION_DESCRIPTOR data structure (see *WDM USB Driver Specification*).

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer, which must be large enough to contain the USB_CONFIGURATION_DESCRIPTOR data structure (see *WDM USB Driver Specification*). Specify zero to return only the length of the Device ID string in lpBytesReturned.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.11.2 Notes

None.

4.11.3 Error Code

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer was not large enough to receive the USB Configuration Descriptor.

4 I/O Control Codes (continued)

4.12 IOCTL_GET_DEVICE_DESCRIPTOR

IOCTL_GET_DEVICE_DESCRIPTOR retrieves the current device descriptor. Details regarding parameters and error codes follow.

4.12.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive a USB_DEVICE_DESCRIPTOR data structure (see *WDM USB Driver Specification*).

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer, which must be large enough to contain the USB_DEVICE_DESCRIPTOR data structure (see *WDM USB Driver Specification*). Specify zero to return only the length of the Device ID string in lpBytesReturned.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.12.2 Notes

None.

4.12.3 Error Code

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer was not large enough to receive the USB Device Descriptor.

4 I/O Control Codes (continued)

4.13 IOCTL_GET_DEVICE_INSTANCES

IOCTL_GET_DEVICE_INSTANCES returns a list of current devices based on the symbolic links created during each device enumeration. Details regarding parameters and error codes follow.

4.13.1 Parameters

lpInBuffer—Points to an input buffer that will receive. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive a DEVICE_INSTANCE_HEADER structure (see Section 5.1) and a series of DEVICE_INSTANCE structures (see Section 5.2), one for each device instance.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Specify zero to return only the length of the Device ID string in lpBytesReturned.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.13.2 Notes

None.

4.13.3 Error Code

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to receive the device instances.

4 I/O Control Codes (continued)

4.14 IOCTL_GET_INTERFACE

IOCTL_GET_INTERFACE gets the information about the current interface and pipes. Details regarding parameters and error codes follow.

4.14.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive a USBD_INTERFACE_INFORMATION data structure (see *WDM USB Driver Specification*).

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer, which must be large enough to contain the USBD_INTERFACE_INFORMATION data structure. Specify zero to return only the length of the Device ID string in lpBytesReturned.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.14.2 Notes

None.

4.14.3 Error Code

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to receive the interface information.

4 I/O Control Codes (continued)

4.15 IOCTL_GET_PORT_STATUS

IOCTL_GET_PORT_STATUS returns a status byte. Details regarding parameters and error codes follow.

4.15.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that receives a byte indicating the current port status. The bit mask is defined as follows (see the *USS-720 USB-to-*IEEE *1284 Bridge* Preliminary Data Sheet, “Get_Port_Status”):

Bit	Meaning
0—2	Reserved, will always read 0.
3	0 = error, 1 = no error.
4	Select.
5	Paper empty.
6—7	Reserved, will always read 0.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. The buffer must be large enough to contain one byte value.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.15.2 Notes

None.

4.15.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—The command is only supported in ALT_INTERFACE_0 and ALT_INTERFACE_1 (see Section 4.9 of this document).

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to receive the interface information.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.16 IOCTL_ISSUE_USS720_COMMAND

IOCTL_ISSUE_USS720_COMMAND issues a specific command for the USS-720 device. Details regarding parameters and error codes follow.

4.16.1 Parameters

lpInBuffer—Points to an buffer that contains a single byte which specifies one of the following USS-720 commands:

Command	Description
CMD_AUTOECP_ON	Turn Auto Mode on.
CMD_AUTOECP_OFF	Turn Auto Mode off.
CMD_COMPRESS_ON	Turn ECP RLE compression on.
CMD_COMPRESS_OFF	Turn ECP RLE compression off.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.16.2 Notes

Returns TRUE if successful; FALSE otherwise.

4.16.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.17 IOCTL_READ_INTERRUPT_PIPE

IOCTL_READ_INTERRUPT_PIPE reports changes on the parallel port and buffer status when they occur. Details regarding parameters and error codes follow.

4.17.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer that will receive four status bytes reflecting changes in the parallel port and buffer status as follows (see the *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, “Registers”):

Byte	Description
0	Status.
1	Control.
2	Extended control.
3	USS-720 control.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer, which must be large enough to contain 4 bytes.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer.

4.17.2 Notes

None.

4.17.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—The command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

ERROR_NOT_ENOUGH_BUFFER—lpOutBuffer needs to be large enough to receive the status bytes.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.18 IOCTL_RESET_PIPE

IOCTL_RESET_PIPE clears the halted state of the specified pipe within the USB stack and resets the stalled state of the endpoint on the device. Details regarding parameters and error codes follow.

4.18.1 Parameters

lpInBuffer—Points to a buffer that contains a single byte which specifies one of the following values:

Value	Description
BULK_OUT_PIPE	Bulk Out pipe.
BULK_IN_PIPE	Bulk In pipe.
INTERRUPT_PIPE	Interrupt pipe

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.18.2 Notes

None.

4.18.3 Error Codes

ERROR_INVALID_PIPE_INDEX—Specified pipe is invalid for this device instance or invalid for the current alternate interface setting.

STATUS_INVALID_PARAMETER—nInBufferSize cannot be zero.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to allocate a request to reset the pipe on this device instance.

4 I/O Control Codes (continued)

4.19 IOCTL_SET_1284_REGISTER

IOCTL_SET_1284_REGISTER sets one of the 1284 registers. Details regarding parameters and error codes follow.

4.19.1 Parameters

lpInBuffer—Points to an input buffer that contains the following 2 bytes:

Byte	Description
0	Value to be written to the register.
1	Address of the parallel port register.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 2.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.19.2 Notes

See *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet, for more detail on the vendor-specific requests SET_1284_REGISTER.

4.19.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—This command is only supported in ALT_INTERFACE_2 (see Section 4.9 of this document).

ERROR_INVALID_ADDRESS_REGISTER—Address of the parallel port register is read only.

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to perform the request on this device instance.

4 I/O Control Codes (continued)

4.20 IOCTL_SET_ALTSETTING

IOCTL_SET_ALTSETTING sets the alternate interface setting. Details regarding parameters and error codes follow.

4.20.1 Parameters

lpInBuffer—Points to an buffer that contains a single byte that specifies one of the following values:

Value	Description
ALT_INTERFACE_0	Bulk Out pipe only.
ALT_INTERFACE_1	Bulk Out and Bulk In pipe.
ALT_INTERFACE_2	Bulk Out, Bulk In, Interrupt pipe, and 1284 register capabilities.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. For this operation, this value should be 1.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.20.2 Notes

None.

4.20.3 Error Codes

ERROR_INVALID_ALTERNATESETTING—The alternate interface setting passed in for this command is invalid.

ERROR_INVALID_DESCRIPTOR—Configuration descriptor is invalid and could not set the alternate interface.

4 I/O Control Codes (continued)

4.21 IOCTL_SOFT_RESET

IOCTL_SOFT_RESET resets the device, flushes the Bulk Out and Bulk In pipes to the default states. Details regarding parameters and error codes follow.

4.21.1 Parameters

lpInBuffer—Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpInBuffer. Not used with this operation. Set to zero.

lpOutBuffer—Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize—Specifies the size (in bytes) of the buffer pointed to by lpOutBuffer. Not used with this operation. Set to zero.

lpBytesReturned—Points to a DWORD that receives the actual size (in bytes) of the data stored into lpOutBuffer. Not used with this operation.

4.21.2 Notes

None.

4.21.3 Error Code

STATUS_INSUFFICIENT_RESOURCES—Not enough memory to allocate a request to perform a soft reset on this device instance.

5 I/O Control Data Structures

This section describes the data structures used in I/O control codes.

5.1 DEVICE_INSTANCE_HEADER

The DEVICE_INSTANCE_HEADER structure is used when issuing the IOCTL_GET_DEVICE_INSTANCES control code.

5.1.1 Data Structure

```
typedef struct _DEVICE_INSTANCE_HEADER{
    DWORD    NumDeviceInstance;
    DWORD    TotalLength;
}DEVICE_INSTANCE_HEADER;
```

5.1.2 Members

NumDeviceInstance—Total number of device instances.

TotalLength—Total length (in bytes) of device instances including the header.

5.2 DEVICE_INSTANCE

The DEVICE_INSTANCE structure is used when issuing the IOCTL_GET_DEVICE_INSTANCES control code.

5.2.1 Data Structure

```
typedef struct _DEVICE_INSTANCE{
    DWORD    InstanceIndex;
    DWORD    LinkNameLength;
    CHAR     DeviceLinkName[64];
}DEVICE_INSTANCE, *PDEVICE_INSTANCE;
```

5.2.2 Members

InstanceIndex—Index of this instance.

LinkNameLength—Length (in bytes) of the DeviceLinkName.

DeviceLinkName—Pointer to a null terminated string containing the symbolic link name of a device instance.

5 I/O Control Data Structures (continued)

5.3 REGISTER_1284

The REGISTER_1284 structure is used when issuing the IOCTL_GET_1284_REGISTER control code.

5.3.1 Data Structure

```
typedef struct _REGISTER_1284
{
    UCHAR RegisterAddress;
    UCHAR DataRegister;
    UCHAR StatusRegister;
    UCHAR ControlRegister;
    UCHAR EppAddressRegister;
    UCHAR EppDataRegister;
    UCHAR EcpCommandRegister;
    UCHAR ExtendedControlRegister;
    UCHAR USS720ControlRegister
}
REGISTER_1284, *PREGISTER_1284;
```

5.3.2 Members

RegisterIndex—Address of the parallel port register.

DataRegister—Data Register value (see Data Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

StatusRegister—Status Register value (see Status Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

ControlRegister—Control Register value (see Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EppAddressRegister—EPP Address Register value (see EPP Address Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EppDataRegister—EPP Data Register value (see EPP Data Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EcpCommandRegister—ECP Command Register value (see ECP Command Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

ExtendedControlRegister—Extended Control Register value (see Extended Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

USS720ControlRegister—USS-720 Control Register value (see USS-720 Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

5 I/O Control Data Structures (continued)

5.4 ADVREGISTER_1284

The ADVREGISTER_1284 structure is used when issuing the IOCTL_GET_1284_REGISTER control code.

5.4.1 Data Structure

```
typedef struct _ADVREGISTER_1284
{
    UCHAR RegisterAddress;
    UCHAR DataRegister;
    UCHAR StatusRegister;
    UCHAR ControlRegister;
    UCHAR EppAddressRegister;
    UCHAR EppDataRegister;
    UCHAR EcpCommandRegister;
    UCHAR ExtendedControlRegister;
    UCHAR USS720ControlRegister;
    UCHAR USS720SetupRegister;
    UCHAR Reserved[10];
}
ADVREGISTER_1284, *PADVREGISTER_1284;
```

5.4.2 Members

RegisterIndex—Address of the parallel port register.

DataRegister—Data Register value (see Data Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

StatusRegister—Status Register value (see Status Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

ControlRegister—Control Register value (see Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EppAddressRegister—EPP Address Register value (see EPP Address Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EppDataRegister—EPP Data Register value (see EPP Data Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

EcpCommandRegister—ECP Command Register value (see ECP Command Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

ExtendedControlRegister—Extended Control Register value (see Extended Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

USS720ControlRegister—USS-720 Control Register value (see USS-720 Control Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

USS720SetupRegister—USS-720 Setup Register value (see USS-720 Setup Register table in Lucent Technologies *USS-720 USB-to*-IEEE *1284 Bridge* Preliminary Data Sheet for details).

Reserved—Reserved for future use.

Application Note
February 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

USS-720 USB Port Monitor

Description

A port monitor is responsible for the communication between the *Windows** spooler and a printing device. It controls the I/O port to which the physical printer is connected and is responsible for the communication channel between the spooler and the print device. Typically, the port monitor communicates with base I/O drivers (e.g., serial and parallel drivers) for device I/O, but it may also call different interfaces, such as *Windows* sockets, SCSI, USB, etc.

Note: For more specifics on the *Windows* spooler and port monitors, see the *Windows* 95/98 DDK documentation.

The USB Port Monitor, developed by In-System Design, enables printing from *Windows* applications using the ISD USB Smart Cable. It is included in the USS-720 Evaluation Kit to demonstrate a functional use of the USS-720 device. Figure 1 shows how the USB Port Monitor is included in the *Windows* spooler.

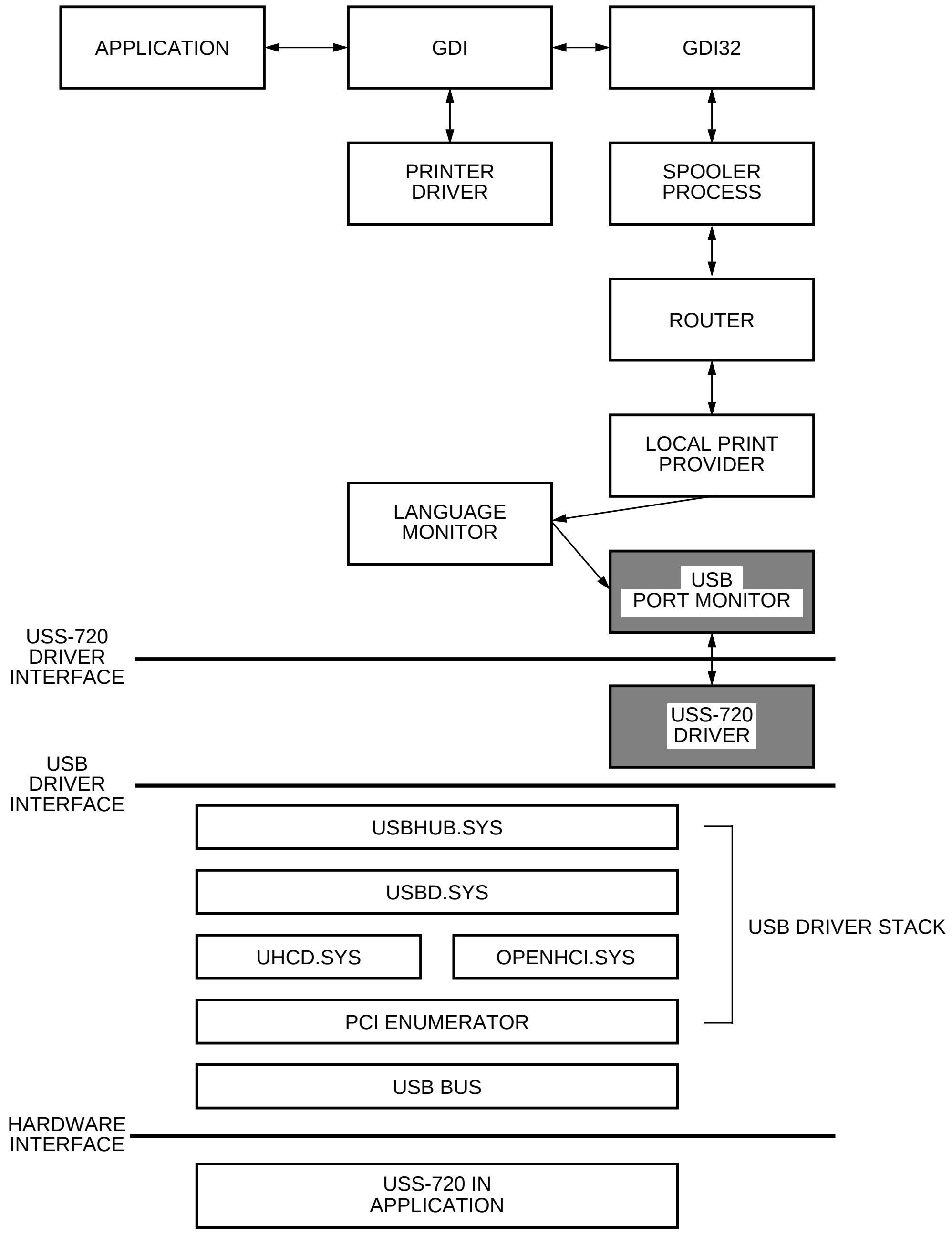
The USB Port Monitor is installed during the Plug and Play installation of the ISD USB Smart Cable. At initialization, the spooler calls the USB Port Monitor to obtain a current list of USB ports available on the host. Once initialization is complete, the spooler has a record of which USB ports exist, and which printers are associated with each USB port. When the spooler determines that a particular spooled job can be printed, it calls the USB Port Monitor to perform the following:

- Notify the start of the print job,
- Send data to be written to the printer,
- Request data to be read from the printer, and
- Notify the end of the print job.

For a description of the data and control flow in and out of the USB Port Monitor, please refer to the Design and Implementation Notes of the *Windows* 95/98 DDK.

* *Windows* is a registered trademark of Microsoft Corporation.

Description (continued)



▨ Software provided with the USS-720

5-6004.r6

Figure 1. Printer Cable with *Instant USB*™

Application Note
June 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

UNIVERSAL SERIAL BUS

Instructions for Downloading Software for Use with the In-System Design USB Smart Cable

Introduction

Before you begin, please note: In order to use the software for the In-System Design USB Smart Cable, the PC where the software will be installed must be running either:

- *Windows** 95/version 4.00.95b with the USB supplement (QFE 1214), also known as OSR2.1 or
- *Windows* 98

No upgrade path is available from previous versions of *Windows* 95/version 4.00.950 to OSR2.1. *Windows* 95/version 4.00.95b (OSR2.0) must first be installed, and then the system can be upgraded to OSR2.1. The USB supplement file, **usbsupp.exe**, is required to upgrade the OSR2.0 operating system to OSR 2.1 (QFE 1214). Contact *Microsoft** to obtain this file.

Software Installation Instructions

1. Point your web browser to **http://www.in-system.com** to access the In-System Design web site.
2. Click on "**Drivers**".
3. For the library name, type: **uss720_dev**

   Note: Use lower-case letters only.
4. For the password, type: **usb_to_lpt**

   Note: Use lower-case letters only.
5. Click on "**Enter Library**".
6. At this point, the uss720_dev library page will appear with a list of files.
7. If your PC is running OSR2.1 with QFE, click on **OSR21_USS720_Eval_Kit_2401.zip** to download the installation files. Continue to step #9 below.
8. If your PC is running *Windows* 98, click on **W98_USS720_Eval_Kit_2402.zip** to download the installation files.
9. Unzip the above file and store the extracted files either on a diskette or in a directory on your hard drive. When you install the USB Smart Cable, you will direct the operating system to search for the files in the location you have selected.

* *Windows* and *Microsoft* are registered trademarks of Microsoft Corporation.

microelectronics group

Lucent Technologies
Bell Labs Innovations

September 1999
MN99-052CMPR-1 (Replaces MN97-061CMPR-04)